



Operator terminals


Dialog 80 and Dialog 640 User guide

SERAD SA

271, route des crêtes
44440 TEILLE – France

 +33 (0)2 40 97 24 54

 +33 (0)2 40 97 27 04

 <http://www.serad.fr>


 info@serad.fr

Table of Contents

1-	INTRODUCTION	4
1-1-	Main presentation	4
1-2-	Dialog 80 presentation	4
1-3-	Dialog 640 presentation	5
1-4-	DWIN software presentation	6
2-	INSTALLATION / STARTING	8
2-1-	Environmental consideration	8
2-2-	Connections	8
2-2-1-	Dialog80	8
A)	Dialog80 connection	8
B)	Dialog80 mounting	9
C)	Mounting of the cell	9
D)	Personnalisable labels	10
2-2-2-	Dialog640	10
A)	Dialog640 connection	10
B)	Dialog640 mounting	11
C)	Mounting of the cell	11
D)	Personnalisable labels	11
3-	DWIN SOFTWARE PRESENTATION	12
3-1-	Software installation	12
3-1-1-	System configuration	12
3-1-2-	Installation procedure	12
3-2-	Architecture	13
3-2-1-	Directories	13
3-2-2-	Project contents	13
3-3-	Presentation	13
3-3-1-	Initial screen	13
3-3-2-	File menu	14
3-3-3-	View menu	16
3-3-4-	Safety menu	16
3-3-5-	Options menu	17
3-3-6-	Help menu	17
4-	TRANSFER A PROJECT	18
4-1-	System setup menu description	18
4-2-	System Boot menu description	18
4-3-	Upgrade from previous version	18
4-4-	Procedure to transfer a project	19
4-5-	Procedure to receive programs	19
4-6-	Procedure to send programs	19
5-	PAGES LIST	21
5-1-	Page	21
5-2-	Declaration of a new page	21
5-3-	Deletion of a page	23
5-4-	Dialog80 editor	23
5-4-1-	Presentation of the editor	23
5-4-2-	Static text	24
5-4-3-	Dynamic text	24
5-4-4-	Numerical value display	26
5-4-5-	Alphanumerical variable display	26
5-4-6-	Numerical variable setting	27

5-4-7- Alphanumerical variable setting	28
5-4-8- Programmation of the simple dynamical keys	29
5-4-9- Programmation of the led function keys	30
5-4-10- Programmation of the ESC key	32
5-5- Dialog640 editor	32
5-5-1- Presentation of the editor	32
5-5-2- Fonts	33
5-5-3- Vertical line display	34
5-5-4- Horizontal line display	34
5-5-5- Rectangle display	34
5-5-6- Change the back screen colour	35
6- DYNAMIC TEXTS	36
6-1- General presentation	36
7- VARIABLES LIST	37
7-1- General presentation	37
7-2- Declaration and modification of a variable	37
7-3- Deletion of a variable	38
8- GLOBAL KEYS COMMANDS	39
8-1- General presentation	39
8-2- Programmation of the dynamic keys	39
8-3- Programmation of the ESC key	41
8-4- Programming the LED function keys	41
9- PROGRAMS	44
9-1- General presentation	44
9-2- Declaration of a program	44
9-3- Deletion of a program	45
10- PASSWORD	46
10-1- General presentation	46
11- ALARMS	47
11-1- General presentation	47
11-2- Adding and configuring an alarm	47
11-3- Cancelling an alarm	48
12- LANGUAGES	49
12-1- General presentation	49
13- PROTOCOLS	50
13-1- Definition of the protocol	50
13-2- MASTER RTU MODBUS	50
13-2-1- Declaration of the ModBus	50
13-2-2- Presentation of the state table	50
13-2-3- Presentation of the command table	52
13-3- CANopen	53
13-3-1- Introduction	53
13-3-2- The CANopen communication	54
13-3-3- Characteristics	55
13-3-4- Dictionary	55
13-3-5- Configuration	56
13-3-6- Example : CANopen link between an operator terminal and a MCS	57

1- INTRODUCTION

1-1- Main presentation

The operator terminal Dialog80 and Dialog640 are used to supervise and to control an installation or a machine which is driven by another peripheral (Numeric comand, PLC, drive, etc...). Each operator terminal have a serial port communication like : RS232 and in option a serial port like : RS422, RS485 or CanBus. The operator terminal uses one of this serial communication link to communicate with a peripheral.

The « intelligent operator terminal » product is the combination of an operator terminal like Dialog80 or Dialog640 associated with the operating system and the DWIN software.

The DWIN software allows to define simply and rapidly on the PC the screen pages of the dialog80 and dialog640 with their links. The definition of the screen pages allows the users to add somme objects like : static or dynamic texts, numeric or alphanumeric display area and numeric or alphanumeric setting area. The Dialog640 have other graphic objects : horizontal or vertical line and rectangle. The transfer of the project is made with a RS232 serial link.

The operator terminal can communicate by using the ModBus or CanBus protocol. The exchange and the type of the exchange of variables is directly configured in the DWIN software.

1-2- Dialog 80 presentation

Screen

- ↪ 4×20 Characters LCD display with backlight
- ↪ Display area 74×23 mm
- ↪ Characters attributes : normal, blinking
- ↪ ASCII protocol

Keyboard

- ↪ 28 keys with tactile feedback
- ↪ 4 dynamic function keys
- ↪ 6 rewriteable function keys with integrated leds
- ↪ Control and scrolling keys
- ↪ Help and alarm keys
- ↪ Numeric and alphanumeric keypad
- ↪ Buzzer

Performances

- ↪ 16 bits Processor
- ↪ 512Kbyte flash memory
- ↪ 128Kbyte non-volatile RAM
- ↪ RS232 serial communication port
- ↪ Optional RS422 or RS485 serial communication port
- ↪ Optional fieldbus : CANBUS

Technical features

- ↵ 24Vdc supply voltage
- ↵ 4W power requirement
- ↵ 0 to 45°C operating temperature
- ↵ -20 to 70°C storage temperature
- ↵ Front panel protection IP65

1-3- Dialog 640 presentation

Screen

- ↵ LCD display with CFL backlight
- ↵ Display area 122×66 mm
- ↵ Characters attributes : normal, reverse, blinking
- ↵ ASCII protocol
- ↵ Resolution in graphic mode : 240×128 pixels
- ↵ 4 simultaneous sizes of characters in text mode :
 - ⇒ 3×4 mm 16 lines × 40 characters
 - ⇒ 4×7 mm 9 lines × 30 characters
 - ⇒ 5×8 mm 8 lines × 26 characters
 - ⇒ 7×10 mm 6 lines × 17 characters

Keypad

- ↵ 33 keys with tactile feedback
- ↵ 6 dynamic function keys
- ↵ 6 rewriteable function keys with integrated leds
- ↵ Control and scrolling keys
- ↵ Help and alarm keys
- ↵ Numeric and alphanumeric keys
- ↵ Buzzer

Performances

- ↵ 16 bits processor
- ↵ 512Kbyte flash memory
- ↵ 128Kbyte non-volatile RAM
- ↵ RS232 serial communication port
- ↵ Optional RS422 or RS485 serial communication port
- ↵ Optional fielbus : CANBUS

Technical features

- ↵ 24Vdc supply voltage
- ↵ 6W power requirement
- ↵ 0 to 45°C operating temperature
- ↵ -20 to 70°C storage temperature

↳ Front panel protection IP65

1-4- DWIN software presentation

The DWIN software is defined with a WYSIWYG (What You See Is What You Get) editor and lots of tools to realize your programming easy. The characteristic of the software are :

↳ Management of 1 to 4 languages for the dialog80 and dialog640

↳ Management of 200 screen pages for the dialog 640 with a maximum adding of 50 objects per page

↳ Management of 600 screen pages for the dialog 80 with a maximum adding of 10 objects per page

↳ Choice of 4 fonts with the normal or inverted mode for the dialog 640

↳ Definition and number of objects :

⇒ 1000 static texts, vertical lines, horizontal lines, rectangles

⇒ 1000 dynamic texts

⇒ 1000 numeric or alphanumeric display area

⇒ 1000 numeric or alphanumeric setting area

↳ Display or setting with the format :

⇒ decimal

⇒ hexadecimal

⇒ binary

⇒ ascii

↳ Management of help pages with the led indicator

⇒ Management of alarm pages with the led indicator :

⇒ Maximum of 320 alarms

⇒ Alarm parameter : with acquit or not, help page associated

↳ Navigation on the different setting area with the arrowed keys

↳ Declaration of local or global keys

↳ Programmable keys :

⇒ push button function

⇒ switch fonction ...

↳ Programmable of the leds :

⇒ switch on

⇒ switch off

⇒ blink

↳ Management of the integrated screen pages :

⇒ Comfirm page

⇒ Access code page

↳ Management of 10 access codes

Dialog80 and Dialog640 operator terminal documentation

- ↳ Management of 3000 static texts for the dialog 80 and 1500 for the dialog 640
- ↳ Management of 3000 dynamic texts for the dialog 80 and 1500 for the dialog 640
- ↳ Management of 5000 variables to exchange with the peripheral :
 - ⇒ word or dword type
 - ⇒ numeric or alphanumeric format
- ↳ Management of 10000 program variables (example : 200 program of 50 elements)

2- INSTALLATION / STARTING

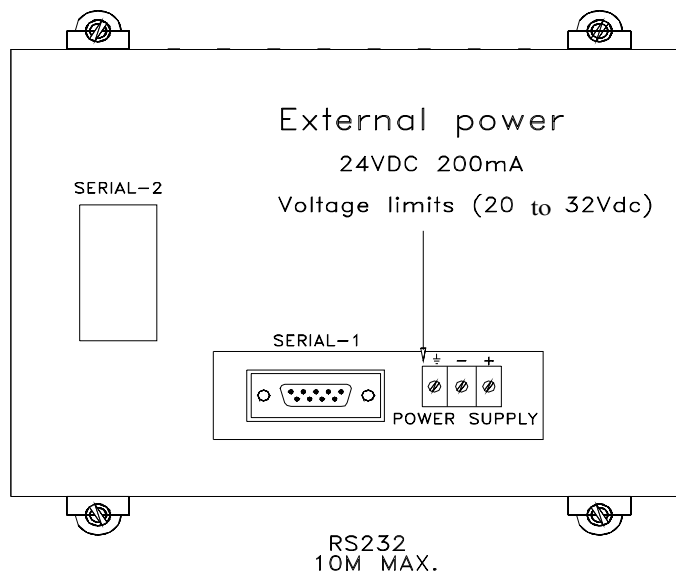
2-1- Environmental consideration

- ↪ 24Vdc supply voltage
- ↪ 6W power requirement for Dialog640 and 4W for Dialog80
- ↪ 0 to 45°C operating temperature
- ↪ -20 to 70°C storage temperature
- ↪ Front panel protection IP65

2-2- Connections

2-2-1- Dialog80

A) Dialog80 connection



SERIAL-1 SUBD 9PTS MALE	
PIN	RS232
1	
2	RXD
3	TXD
4	
5	GND
6	
7	
8	
9	

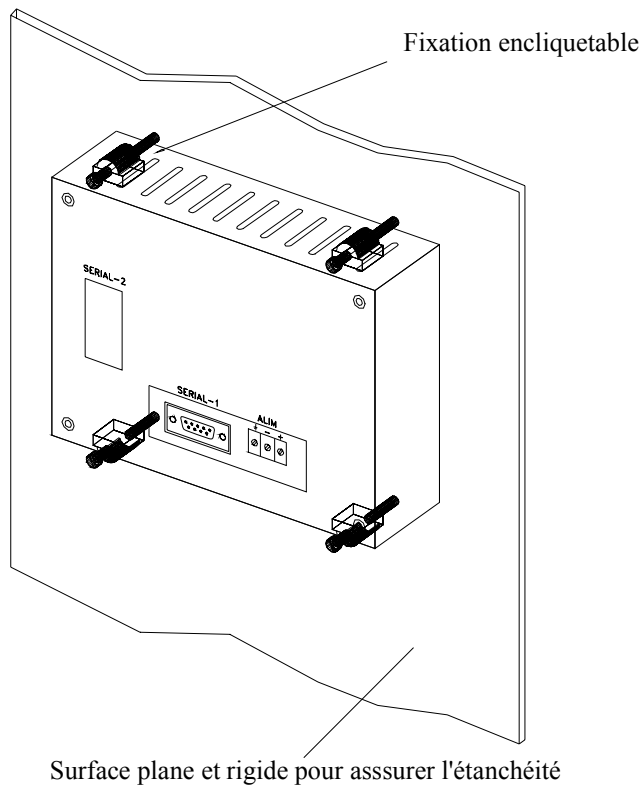
NON ISOLATED

OPTIONAL SERIAL-2 SUBD 9PTS FEMALE			
PIN	RS422	RS485	CANBUS
1			
2			
3	RX -		
4	RX +		
5	GND	GND	GND
6			
7	TX -	TRX -(B)	CANL
8	TX +	TRX +(A)	CANH
9			

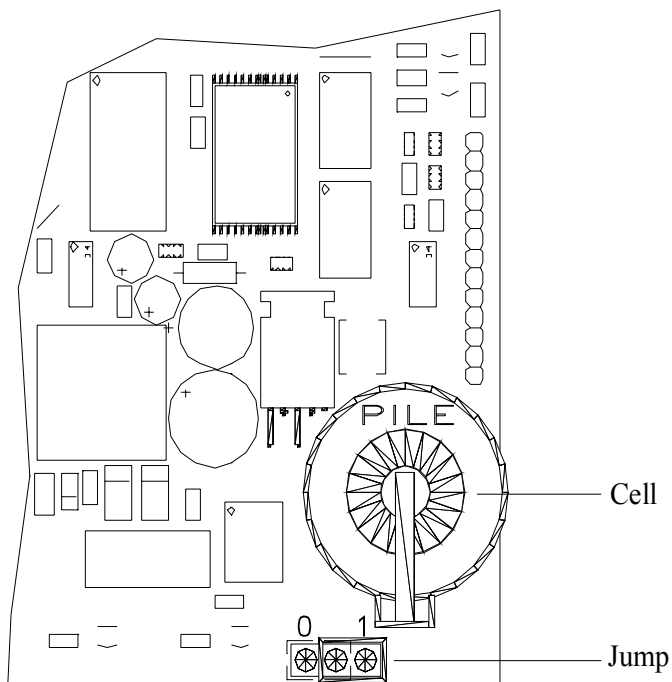
ISOLATED

On the RS422 or RS 485 modules : Jump of validation for final resistors (120Ω). By default : selected resistors.

B) Dialog80 mounting



C) Mounting of the cell



Cell CR2450 : 3V - 500mA

- ↪ Open the back of panel
- ↪ Insert the cell in its seating – Verify polarity
- ↪ Put jump position 1

Warning : Version without cell : jump position 0

D) Personalisable labels

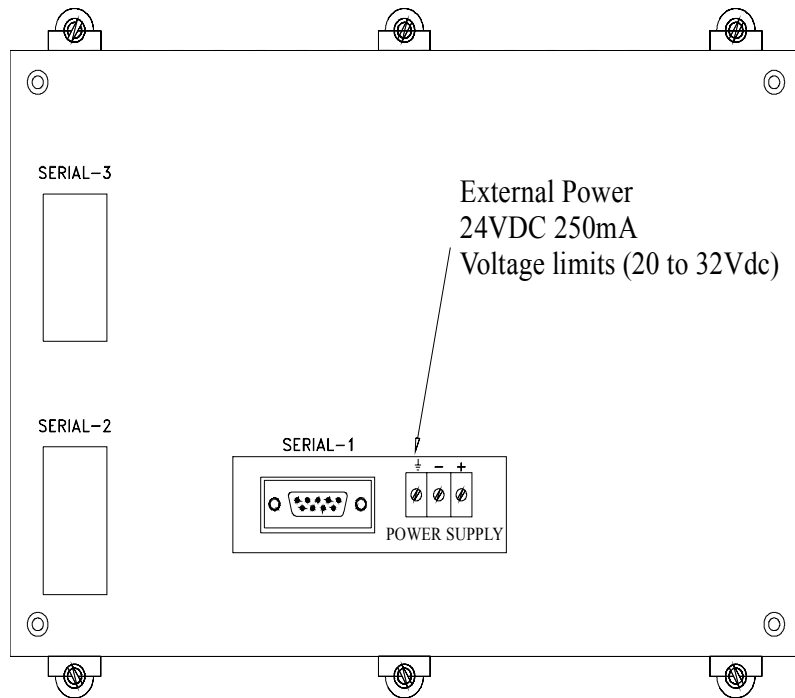
The F1 to F6 keys can be customized. You can use customized blank labels supplied and write new informations.

- ↪ Open the back of panel
- ↪ Remote label already placed down left
- ↪ Introduce new label and close behind panel

Warning : Turn off all power supply of this equipement

2-2-2- Dialog640

A) Dialog640 connection



LIASON RS232
L=10M MAX.

OPTIONAL

SERIAL-1 SUBD 9PTS MALE	
PIN	RS232
1	
2	RXD
3	TXD
4	
5	GND
6	
7	
8	
9	

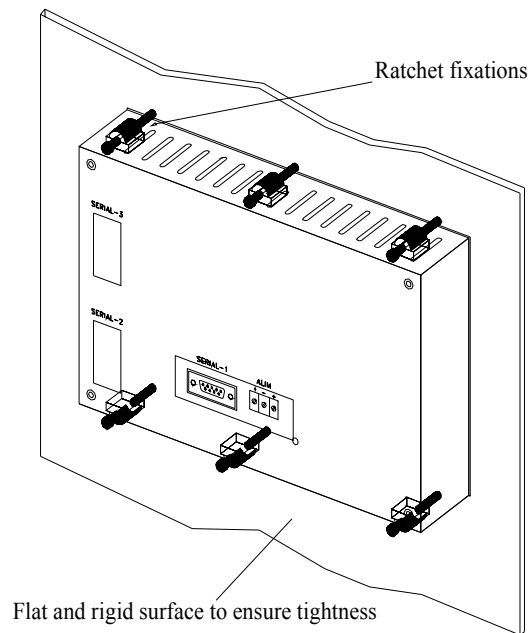
NON ISOLATED

SERIAL-2 SUBD 9PTS FEMALE			
PIN	RS422	RS485	SERIAL-3 SUBD 9PTS FEMALE
			CANBUS
1			
2			
3	RX -		
4	RX +		
5	GND	GND	GND
6			
7	TX -	TRX -(B)	CANL
8	TX +	TRX +(A)	CANH
9			

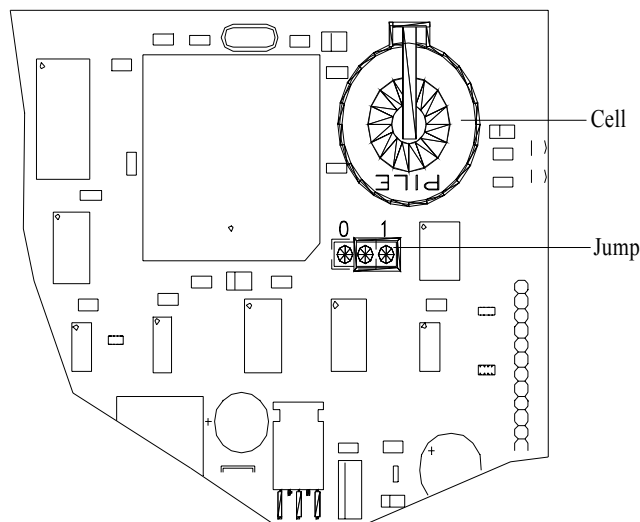
ISOLATED

On the RS422 or RS 485 modules : Jump of validation for final resistors (120Ω). By default : selected resistors.

B) Dialog640 mounting



C) Mounting of the cell



Cell CR2450 : 3V - 500mA

- ↪ Open back of the panel
- ↪ Insert the cell in its seating – Verify polarity
- ↪ Put jump position 1

Warning : Version without the cell : jump position 0

D) Personalisable labels

The F7 to F12 keys can be customized. You can use customized blank labels supplied and write new informations.

- ↪ Open the back of panel
- ↪ Remote label already placed down left
- ↪ Introduce new label and close behind panel

Warning : Turn off all power supply of this equipement

3- DWIN SOFTWARE PRESENTATION

3-1- Software installation

3-1-1- System configuration

Minimal configuration :

- ↵ PC 486 DX2 66
- ↵ RAM 8 Mb
- ↵ Hard disk (15 Mb available)
- ↵ Microsoft® Windows™ 95 or Microsoft® Windows™NT 4.0 (service pack 3)
- ↵ CD-ROM (2X)
- ↵ SVGA colour display
- ↵ Mouse or other peripheral pointing system


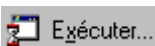
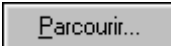



Required configuration :

- ↵ PC Pentium® 75 or greater
- ↵ RAM 16 Mb
- ↵ Hard disk (15 Mb available)
- ↵ Microsoft® Windows™ 95 or Microsoft® Windows™NT 4.0 (service pack 3)
- ↵ CD-ROM (4X)
- ↵ SVGA colour display
- ↵ Mouse or other peripheral pointing system

This software run on Microsoft® Windows NT™. But, it doesn't run on Unix, Mac, MS-DOS and Microsoft® Windows 3.11.

3-1-2- Installation procedure

The DWIN software is provided on floppy disk (if required) or in a CD-ROM with the Dialog80 or Dialog640 terminal operator. The installation procedure is described below :

- ↵ Verify the required configuration before the software installation
- ↵ Insert the floppy disk or the CD-ROM in the appropriate drive.
- ↵ In the menu , select  .
- ↵ In the « Execute » dialog box , select  .
- ↵ In the « Parcourir » dialog box, select the drive where the floppy disk or CD-ROM is.
- ↵ Select  Setup.exe then  in the « Parcourir » dialog box.
- ↵ Select  in the « Execute » dialog box.
- ⇒ The installation software of DWIN is running.
- ↵ In the beginning of the installation, there are some dialog box to drive the installation :
 - Destination folder

- Installation type (Typical, compact or custom)
- Select the program manager
 - ⇒ Warning : only one level of folder can be created.
 - ⇒ **The file installation starts and is indicated by the evolution of a progress bar.**
 - ⇒ **The installation finishes with the adding of DWIN icon in the program manager.**

3-2- Architecture

3-2-1- Directories

- ↳ OS/Dialog80 : contains a copy of Dialog80 terminal operator operating system.
- ↳ OS/Dialog640 : contains a copy of Dialog640 terminal operator operating system.
- ↳ HELP : contains the help files of the DWIN software.

On the main directories, we can have :

- ↳ the file with EXE extension to run the software
- ↳ the file with STR extension to manage languages

3-2-2- Project contents

There is no specific directories to store the user's projects. Users must define a directory for each project. Each project have a name file like "Project Name.otf". The compiled project makes some binary files (Project Name.da0 to Project Name.da7). The sum of all da* type files is the length of the compiled project.

3-3- Presentation

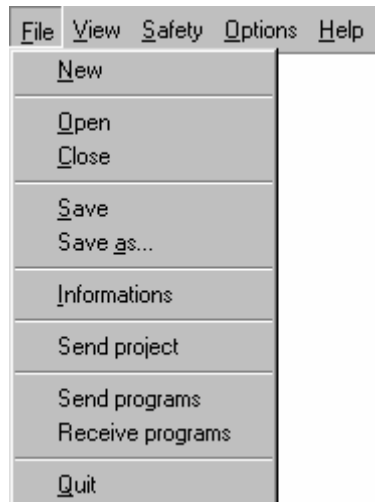
3-3-1- Initial screen




The main window of the DWIN software has a menu, an icon zone and a working zone where windows can appear. The menu is defined like this :

- ↳ File : regroups all the project commands (new, save, close...) and the command for the project transfer.
- ↳ View : regroups all the commands to display of variables list window, pages list window, etc...
- ↳ Safety : regroups all the commands to display of the access code list window and alarms list window.
- ↳ Options : regroups all the commands of the software and terminal operator parameters
- ↳ Help : regroups all the help commands for the software.

3-3-2- File menu



New

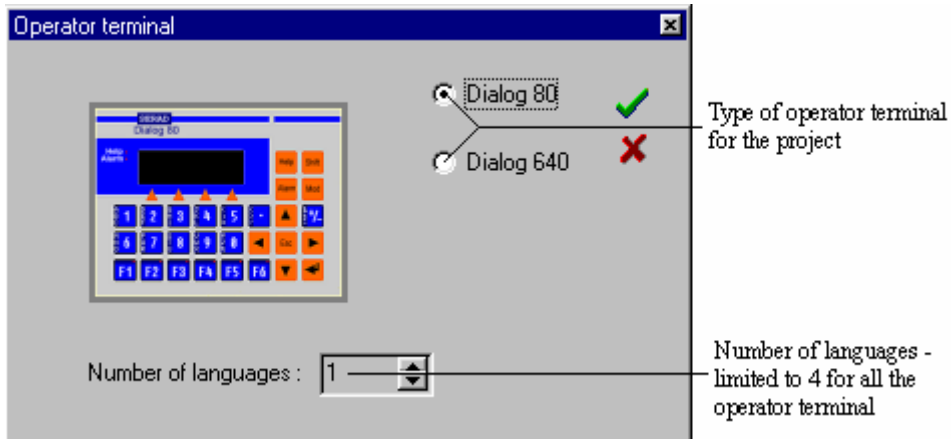
Icon : 

Action : This commande defines a new project. The last open project is closed. A dialog box allows the configuration of the necessary parameters :

↳ the destination operator terminal (dialog80 or dialog640)

↳ the number of languages which the terminal operator can manage (1 to 4)

This parameters are important because we can never change them at any other time.



Open

Icon : 

Action : This command opens the dialog box « Open a project ». The user can specify its directory and the name of the file to open (OTF extension). The last open project is closed when the new project is opened.

Close

Action : This command closes the current project.

Save

Icon : 

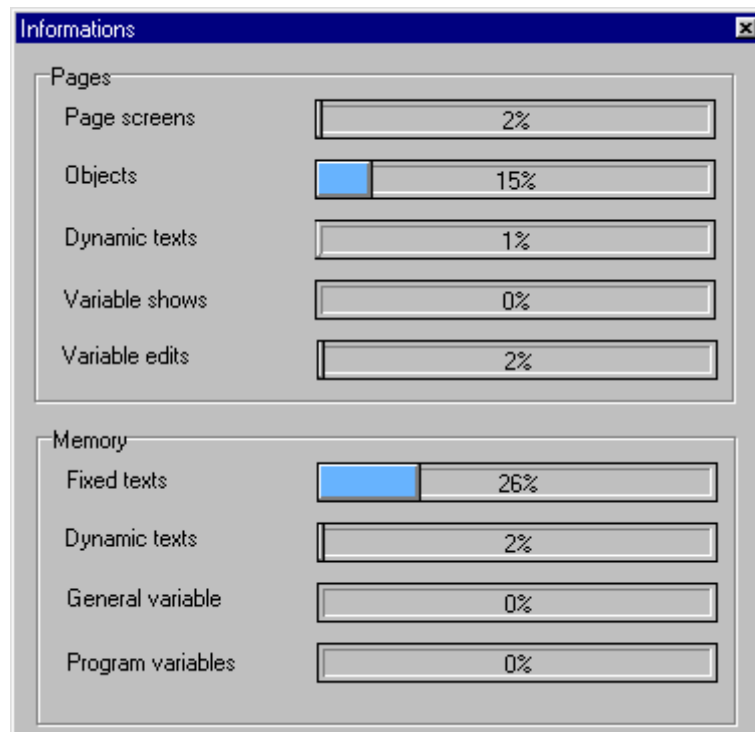
Action : This command saves the current project.

Save as...

Action : This command allows the modification of the name of the project file and of its directory.

Informations

Action : This command opens a window which indicates the resources used by the current project.



In the example above, these are the informations of a dialog640 project. We can see that 2% of the screen pages are used by this project on a maximum of 200.

Send project

Action : This command sends the compiled project to the operator terminal. To realize this operation, you must follow the instruction of the transfer procedure.

Send programs

Action : This command sends a backup of the programs to the operator terminal. The project on the operator terminal and on the PC must be the same. Before the beginning of the transfer, you must define the communication port. The sending programs procedure is described in a specific chapter.

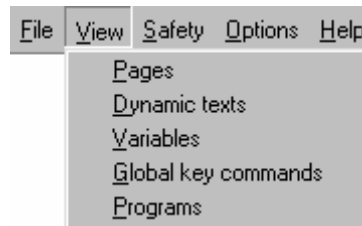
Receive programs

Action : This command receives the data of the operator terminal programs. To have a backup of this programs, you must save the project. The project on the operator terminal and on the PC must be the same. Before the beginning of the transfer, you must define the communication port. The receiving programs procedure is described in a specific chapter.

Quit






Action : This command stops the execution of DWIN software.

3-3-3- View menu

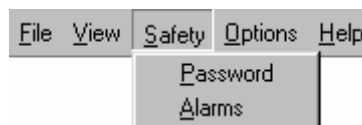


Action : These commands validate or not the viewing of windows like : Pages list, dynamic texts list, variables list, Global keys command or programs

These commands can also be obtain with these icons :


- ↪ Pages : 
- ↪ Dynamic Texts : 
- ↪ Variables : 
- ↪ Global keys command : 
- ↪ Programs : 

3-3-4- Safety menu

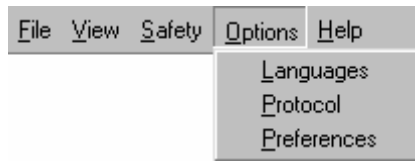


Action : These commands validate or not the viewing of windows like : Access code list or Alarms list.


These commands can also be obtain with these icons :

- ↪ Password : 
- ↪ Alarms : 

3-3-5- Options menu



Languages

Icon : 

Action : This command activates or not the viewing of the languages management window.

Protocol

Action : This command opens the communication protocol dialog box. The protocols, which can be define between the operator terminal and peripheral device, are : ModBus, CanOpen.

Préférences

Action : This command allows the modification of DWIN software parameters. The parameters are the language used by DWIN software and the backlight delay of the dialog640.

3-3-6- Help menu

About

Action : This command indicates the software version.

Index

Action : This command launches the DWIN help on line.

4- TRANSFER A PROJECT

4-1- System setup menu description

To have the setup menu of the operator terminal, you need to press simultaneously the ESC and RETURN key at anytimes. The access code depends of the operator terminal used. It is 80 for a dialog80 and 640 for a dialog640.

The setup menu has a PC menu which allows the transfer of project and datas, an INFO menu which gives some operator terminal configuration information and a QUIT command to return to the normal running mode.

The PC menu allows to load project (LOAD PROJECT), to load programs (LOAD DATA) and to save programs (SAVE DATA). Each command of this menu is detailed in a specific chapter.

The INFO menu has the project information command (PROJECT), the operating system version command (OS) and the serial link configuration command (SERIAL). The project information are : the name, the loading date of the project and the size of the project. The operating system information are : the version number and its date of loading. The serial link informations are : the type, the configuration and the protocol used.

4-2- System Boot menu description

To have the boot menu of the terminal operator, you need to press the ESC key before and to turn on the power supply. The access code depends of the operator terminal used. It is 80 for a dialog80 and 640 for a dialog640.

The PC menu has the operating system upgrading command. The procedure to upgrade the operating system is described in a specific chapter (Upgrade from previous version). The TEST command allows the test of operator terminal keys and leds. The INFO command gives the version of the operating system and its date of loading.

4-3- Upgrade from previous version

↳ To upgrade the operating system, you need to link the operator terminal and the PC with a serial link. Only the serial port 1 (COM1) of the PC is allowed.

↳ You need to access the boot menu of the operator terminal. To realize this, you must press the ESC key before and turn on the power supply. The access code depends of the operator terminal used. It is 80 for a dialog80 and 640 for a dialog640.

↳ You select the PC menu with the 2 key of the dialog80 or with the F1 key of the dialog640.

↳ To this menu, you choose the LOAD command with the 2 key of the dialog80 and with the F1 key of the dialog640. The terminal operator erases its different memory blocs before waiting the receiving of the new operating system. It signals its waiting by the message « Waiting for program ».

↳ On the PC, you must take place in the OS directory of the main directory of DWIN (C:\Program Files\Serad\Operator terminal). In this directory, you must execute the Install.bat file.

↳ A DOS window is opened and wait a key pressed before to start the sending of operating system.

↳ When the transfer is finished, you need to press a key to quit the program on PC. So, you can close the DOS window. It's necessary to execute this procedure if you want the software lets the serial communication port free.

↵ The operating system is now ready

4-4- Procedure to transfer a project

↵ To send a project to a terminal operator, you need to link the operator terminal and the PC with a serial link. The communication port of the PC can be any of them.

↵ You need to access the setup menu of the operator terminal. To realize this, you must press simultaneously the ESC key and the RETURN key at any times. The access code depends of the operator terminal used. It is 80 for a dialog80 and 640 for a dialog640.

↵ You choose the PC menu with the 2 key of the dialog80 or the F1 key of the dialog640. Then, you choose the LOAD PROJECT command with the 2 key of the dialog80 or the F1 key of the dialog640. You validate the command (YES) with the 2 key of the dialog80 or the F1 key of the dialog640. The terminal operator begins to erase its memory and waits the sending of project. Its signals its waiting with the message : « Waiting for project ».

↵ In the DWIN software, you must select the « Send project » command of the « File » menu. You choose the serial PC port linked to the operator terminal.

↵ The PC shows the progress of the transfer. When the progress bar disappears, it signals that the transfer is finished. The operator terminal boots up and executes the loaded project.

4-5- Procedure to receive programs

↵ To save the programs of a operator terminal, you need to link the operator terminal and the PC with a serial link. The communication port of the PC can be any of them.

↵ In DWIN, you need to open the project corresponding to this one on the operator terminal.

↵ You need to access the setup menu of the operator terminal. To realize this, you must press simultaneously the ESC key and the RETURN key at any times. The access code depends of the operator terminal used. It is 80 for a dialog80 and 640 for a dialog640.

↵ You choose the PC menu with the 2 key of the dialog80 or the F1 key of the dialog640. Then, you choose the SAVE DATA command with the 5 key of the dialog80 or the F6 key of the dialog640. The terminal operator waits the request of the PC before to start. Its signals its waiting with the message : « Waiting for request ».

↵ In the DWIN software, you must select the « Receive programs » command of the « File » menu. You choose the serial PC port linked to the operator terminal.

↵ The PC shows the progress of the transfer. When the progress bar disappears, it signals that the transfer is finished. The operator terminal boots up and executes the project.

↵ It's necessary to save the project on the PC.

4-6- Procedure to send programs

↵ To backup the programs of a terminal operator, you need to link the operator terminal and the PC with a serial link. The communication port of the PC can be any of them.

↵ In DWIN, you need to open the project corresponding to this one on the operator terminal.

↵ You need to access the setup menu of the operator terminal. To realize this, you must press simultaneously the ESC key and the RETURN key at any times. The access code depends of the operator terminal used. It is 80 for a dialog80 and 640 for a dialog640.

↵ You choose the PC menu with the 2 key of the dialog80 or the F1 key of the dialog640. Then, you choose the LOAD DATA command with the 4 key of the dialog80 or the F5 key of

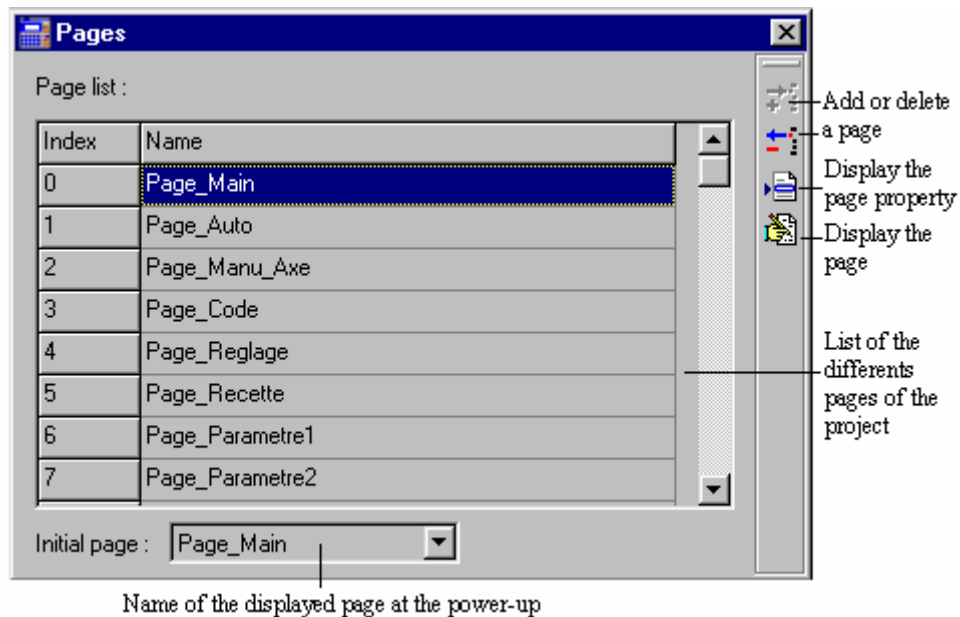
the dialog640. The terminal operator waits the sending of data from the PC. Its signals its waiting with the message : « Waiting for data ».



↳ In the DWIN software, you must select the « Send programs » command of the « File » menu. You choose the serial PC port linked to the operator terminal.

↳ The PC shows the progress of the transfer. When the progress bar disappears, it signals that the transfer is finished. The operator terminal boots up and executes the project.



5- PAGES LIST

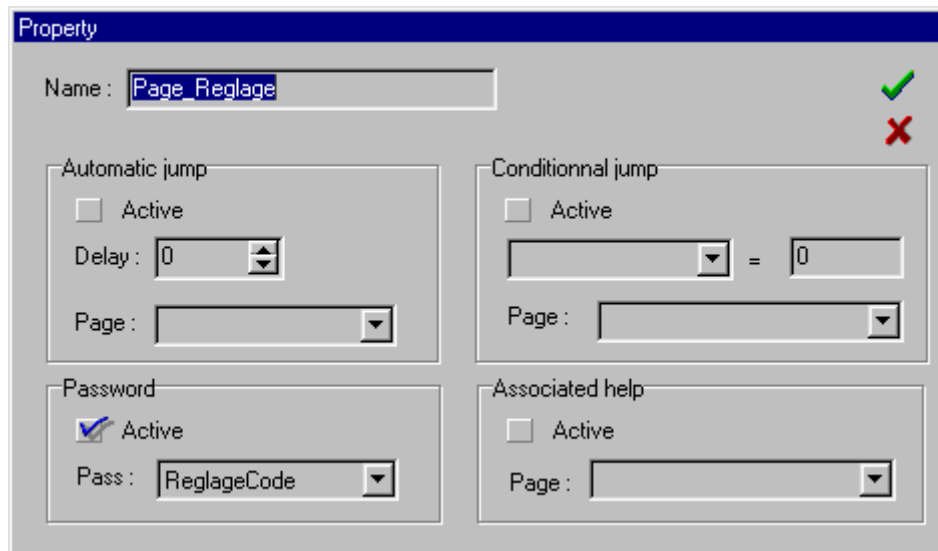
5-1- Page



This window shows the pages list of a project. It can be activated or not with the icon . It is from this window that we can have an action on the screen pages, with the declaration, the suppression, the properties modification, the pages edition. On this page, we can declare the initial screen, which will be displayed for any power-on (at least one screen must be declared in the list). To edit a page, we must choose the command . The editor is different regarding the type of the terminal: one is dedicated to the Dialog 80, and the other one to the Dialog 640. We can also go to the editor with a left double-click on a declared page place.

5-2- Declaration of a new page

The definition of a new page is obtained either with a left double-click on an empty place or a left click on the icon  of the screens list window. Activating this command makes the page property window appears. It is on this window that we can configure the new page specifications. This window can also be obtained for an already existing screen, when you select the screen before clicking with the left button on the icon .



In the property window, we can define or modify the screen name. There is no restriction on the screen name.

The screen properties are mainly its links with the others. To take in account any property, the associated box must be ticked.

↳ **Automatic jump** : This property makes an automatic jump to another screen after a delay of displaying the current one. The parameters to define are:

- ⇒ the delay, expressed in second
- ⇒ the name of the page to which is made the jump

Example : `delay=2s ; Page= NextPage`
 A jump to NextPag will be done after being 2s in the CurrentPage.

↳ **Conditional jump** : This property makes a jump to another page when a equality condition is true. This link is done only if the current page is displayed on the terminal and if the equality condition is true. The test is an equality between a variable defined in the variables list and a constant value. The parameters to define are:

- ⇒ the name of the variable to test
- ⇒ the constant value for the test
- ⇒ the name of the page to which is made the jump

Example: `equality: var=4 ; Page= NextPage`
 A jump to NextPage will be done when Var is equal to 4 while the CurrentPage is displayed.

↳ **Password** : This property displays a page which is asking for a password before displaying the current page. The current page will be displayed only if the keyed code is correct. In the other way, the password page will stay. The password page is a pre-defined system page, so you do not have to define it. The parameters to define are:

- ⇒ the name of the variable password

Example : `Pass=PagePass ;`
 A jump to the CurrentPage will be done when the correct password is keyed.


↳ **Associated help** : This property permits to define if a help page is available with the current one. During the display of the page on the terminal, the association of an help page is shown with the flashing of the LED Help. Then we can go to the help page by pressing the key Help on the terminal. The parameters to define are:

- ⇒ the help page

Example : `Page= HelpPage`

A jump to HelpPage will be done if you press the Help key while the CurrentPage is displayed. You can also see the LED flashing during this display, which tells that a help page is available.

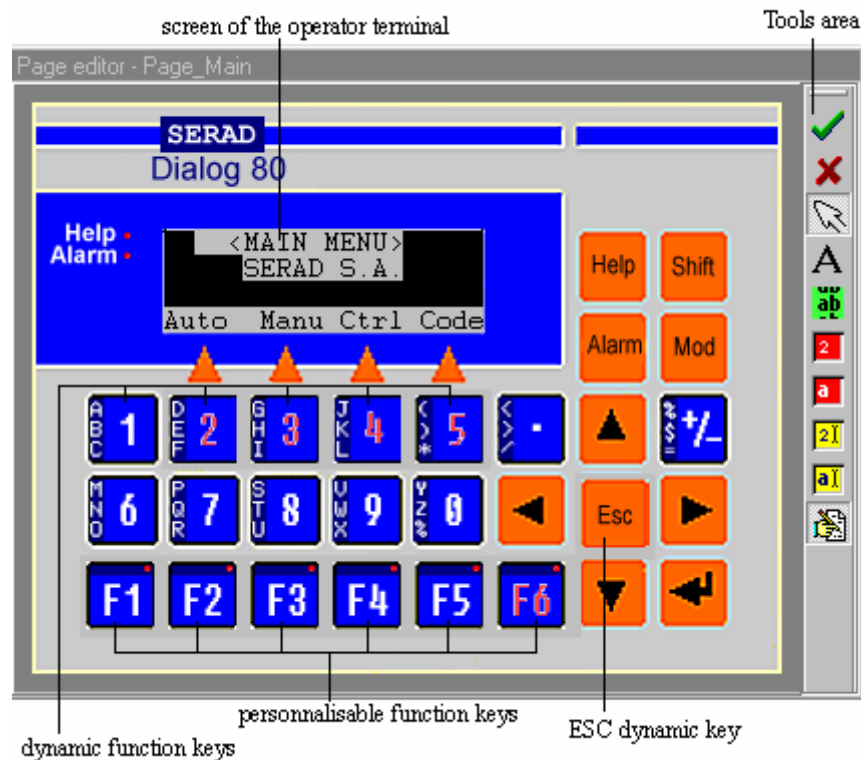
5-3- Deletion of a page

The deletion of a page is done by selecting the page to delete, and then by a left click on the icon . The deletion does not shift the upper pages, it lets an empty place in the list, which can be used by another page. If the deleted page is used as a link for other pages, their fields become empty. The user must think on re-define them.

5-4- Dialog80 editor



5-4-1- Presentation of the editor







The editor is made of 2 windows.




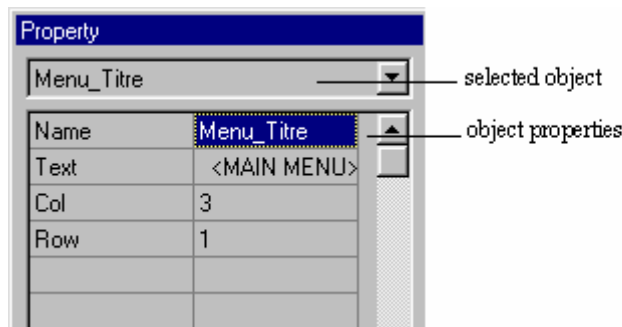
The first window is the full screen editor. In this one, we can see the front page of the Dialog 80 and a tools bar on the right. On this screen, we can configure:


- ↳ the keys 2, 3, 4, 5 of the terminal with a left click on them
- ↳ the keys F1, F2, F3, F4, F5, F6 and their leds with a left click on the corresponding key.
- ↳ the ESC key with a left click on it
- ↳ the terminal screen

The tools bar is made of the icons: page edition validation , page edition cancelling . It is also made of these icons, to add objects on the terminal screen:

- ⇒ Static text 
- ⇒ Dynamic text 
- ⇒ Numerical variable display 
- ⇒ Alphanumerical variable display 
- ⇒ Numerical variable setting 
- ⇒ Alphanumerical variable setting 


The icon  can position on a screen object to modify its size or its properties. This icon is automatically as an object is added on the screen.



The last icon  can validate or not the second window which concerns the properties of the objects on the terminal screen. In this window, you can select the desired object defined with its name and display its properties. You can also select an object with a left click on this object on the terminal screen. The list of the properties on the left is different regarding the type of the selected object. The fields of the right column are the characteristics of the parameters of the left column, so you can modify them. Any modification of these parameters is shown by a modification of the object on the screen. In the same way, any modification of the object on the screen modifies its properties.

5-4-2- Static text

This kind of object displays a fixed text on the screen. This text can be defined by the user.


- ⇒ The adding of this object is made by selecting the icon  and by a left click on the terminal screen.
- ⇒ When you click on the object while keeping the left button pressed, you move the static text following the mouse.

The properties of this object are:

- ⇒ **Name** : It defines the object name, and must be unique.
- ⇒ **Text** : It defines the printed text. The object length is automatically adapted to the text length. If the text is bigger than the screen size, then the characters too many will not be printed.
- ⇒ **Col** : It defines the first column of the screen where the text will be located.
- ⇒ **Row** : It defines the line of the screen where the text will be located.

5-4-3- Dynamic text

This kind of object displays a dynamic text on the screen. This text is depending on an index variable, which points to a text from the dynamic texts list.

⇒ The adding of this object is made by selecting the icon  and by a left click on the terminal screen.

⇒ When you click on the object while keeping the left button pressed, you move the dynamic text following the mouse.

⇒ When you click on the right or left edge of the object and keep the left button on, you can modify the length of the displayed text.

The properties of this object are:

↵ **Name** : It defines the object name, and must be unique.

↵ **Col** : It defines the first column of the screen where the text will be located.

↵ **Row** : It defines the line of the screen where the text will be located.

↵ **Length** : This property specifies the object length, and so the allowed characters number. If the text to display is too long, then the characters too many will not be printed.

↵ **Index** : It defines the starting address of the dynamic list when the variable coding is on a type BCD or binary. When this value is zero, the starting address is directly the low index. When the variable coding is on a type bit, this property indicates the number of the bit which will be used as an index. For example, if the value is 4 with a BCD or binary coding type, the dynamic texts will be displayed from the address 4. If the coding type is bit, the display will depend on the value of the fourth bit of the variable (LSB is bit 1).

↵ **Bit 0** : It is valid only if the property coding is a bit type. It defines the index value when the bit is reset. If we key 25, the text displayed for a 0 bit's value is the one located at the index 25 of the dynamic texts list.

↵ **Bit 1** : It is valid only if the property coding is a bit type. It defines the index value when the bit is set. If we key 26, the text displayed for a 1 bit's value is the one located at the index 26 of the dynamic texts list.

↵ **Number** : It is valid only if the property coding is a binary or BCD type. It defines the maximum text number that can display the object. Then we can know the maximum index value for this object, which will be Index + Number-1.

↵ **Variable** : It defines the address variable which determines the text to display in the dynamic list. The displayed text will correspond to the address Index + variable value, with this address includes from Index and Index + Number - 1.

↵ **Coding** : It defines the coding type of the variable, which may come from any peripheral. Properties are different referring to the coding type. A bit type can have only two states, so we can only display a dynamic text from two. Associated properties are: bit0, bit1 and number. The BCD type means that the evolution of the variable is made with a BCD format: for example, 01010011b is 93. The binary type means that the evolution of the variable is made with a binary format: for example, 01010011b is 83. The associated properties for BCD and binary are number and index.

Example 1:

```
Coding=binary ; Index=5 ; Number=15 ; Variable=VarTextDyn
If contents of VarTextDyn=0h, then display of dynamic text n°5
If contents of VarTextDyn=1h, then display of dynamic text n°6
Etc.
If contents of VarTextDyn=0Ah, then display of dynamic text n°15
Etc.
If contents of VarTextDyn=0Eh, then display of dynamic text n°19
If contents of VarTextDyn>0Eh or <0, then no display
```


Example 2:

```
Coding=BCD ; Index=5 ; Number=15 ; Variable=VarTextDyn
The format of the variable VarTextDyn must be BCD.
```

If contents of VarTextDyn=0h, then display of dynamic text n°5
If contents of VarTextDyn=1h, then display of dynamic text n°6
Etc.
If contents of VarTextDyn=10h, then display of dynamic text n°15
Etc.
If contents of VarTextDyn=14h, then display of dynamic text n°19
If contents of VarTextDyn>14h or <0, then no display
Example 3:
Coding=Bit ; Number=5 ; Bit0=55 ; Bit1=32 ; Variable=VarText
If contents of VarTextDyn=XXX0XXXXb, then display of dynamic text n°55
If contents of VarTextDyn=XXX1XXXXb, then display of dynamic text n°32

5-4-4- Numerical value display

This kind of object can display the value of a numerical variable on the screen.

- ⇒ To add an object, choose the icon  and have a left click on the terminal screen.
- ⇒ When you click on the object while keeping the left button pressed, you move the numerical value following the mouse.


The properties of this object are:

- ↵ **Name** : It defines the object name, and must be unique.
- ↵ **Col** : It defines the first column of the screen where the variable will be located.
- ↵ **Row** : It defines the line of the screen where the variable will be located.
- ↵ **Variable** : It defines the variable to display.
- ↵ **Sign** : It defines if the sign of the variable must be displayed.
- ↵ **Integer part** : It defines the number of figures displayed before the decimal point.
- ↵ **Decimal part** : It defines the number of figures displayed after the decimal point.
- ↵ **Align** : It defines the justification of the number regarding its place (right or left).
- ↵ **Base** : It defines the type of base used to display the variable (Decimal, hexadecimal, binary)

Example :
Base=Decimal ; Sign=yes ; Integer part=3 ; Decimal part=2 ;
Variable=VarNum
If contents of VarNum=+3.25, then display is +3.25
If contents of VarNum=-2255.25, then display is -2255.2
If contents of VarNum=+255.2564, then display is +255.25

5-4-5- Alphanumerical variable display

This kind of object can display the value of an alphanumerical variable on the screen.

- ⇒ The adding of this object is made by selecting the icon  and by a left click on the terminal screen.
- ⇒ When you click on the object while keeping the left button pressed, you move the alphanumerical variable following the mouse.
- ⇒ When you click on the right or left edge of the object and keep the left button on, you can modify the length of the displayed text.

The properties of this object are:

- ↵ **Name** : It defines the object name, and must be unique.
- ↵ **Col** : It defines the first column of the screen where the text will be located.
- ↵ **Row** : It defines the line of the screen where the text will be located.
- ↵ **Variable** : It defines the variable to display.

↵ **Length** : This property indicates the maximum characters numbers to display

5-4-6- Numerical variable setting

This object can define a field to input a numerical variable on the screen.

⇒ To add an object, choose the icon  and have a left click on the terminal screen.

⇒ When you click on the object while keeping the left button pressed, you move the numerical variable following the mouse.

The properties of this object are:

↵ **Name** : It defines the object name, and must be unique.

↵ **Col** : It defines the first column of the screen where the text will be located.

↵ **Row** : It defines the line of the screen where the text will be located.

↵ **Variable** : It defines the variable to display and to affect the input value.

↵ **Help** : It indicates if a help menu is available with this input. If it is on Yes, it means that when the navigator “>” is located on this object, the Help led flashes and the associated help page can be displayed with the Help key.

↵ **Help Page** : It indicates the name of the page which will be as an help. To display it, the Help property must be true.

↵ **Password** : It indicates if an access code must be specified when modifying the field value (as soon as the Mod key is pressed).

↵ **Pass name** : It indicates what will be the access code of the codes list which will be needed.

↵ **Sign** : It defines if the sign of the variable must be displayed.

↵ **Integer part** : It defines the number of figures displayed before the decimal point.

↵ **Decimal part** : It defines the number of figures displayed after the decimal point.

↵ **Align** : It defines the justification of the number regarding its place (right or left).

↵ **Base** : It defines the type of base used to display the variable (Decimal, hexadecimal, binary)

↵ **Limit** : It indicates if limits must be applied to the input value.

↵ **Min limit** : It is the minimum allowed input value.

↵ **Max limit** : It is the maximum allowed input value.

↵ **Navig** : It indicates the scheduling of the different input fields. The navigator “>” shows which field will be affected on the Mod key-pressed. To pass the indicator “>” from one field to the other, you just have to use the up and down arrows. To schedule the navigator, we indicate a number in the parameter “Navig”. If this value is 1, the indicator will be at this place each time we arrive on this page. A down arrow key-pressed can pass to the field with the “Navig” parameter equal to 2, etc. To come back, you just have to press the up arrow key. All the “Navig” parameters on a page must be different. If 2 are the same, only one will receive the navigator “>” and be able to be modified. In the same way, the zero value in that field is not allowed.

Example :

Creation scheduling of the different inputs fields of a page

Input field n°1 : Navigator=2

Input field n°2 : Navigator=1

Input field n°3 : Navigator=3

Input field n°4 : Navigator=0

Input field n°5 : Navigator=3

Input field n°6 : Navigator=8

When the page is displayed, the indicator is on the input field number 2. A

down arrow key-pressed changes it to the number 1, another one to the number 5. Actually, fields number 3 and 5 have the same property, but only the last input field will be addressed by the navigator. A new down arrow key-pressed goes to number 6. The up arrow key-pressed passes in the opposite way. The field number 4 will never have the indicator.

☞ **Confirm** : It indicates if a confirmation screen must appear after the input setting (ENTER key-pressed). The confirmation screen does not need to be written by the user. To confirm, you have to press the key 2, and the key 3 for cancelling.

5-4-7- Alphanumerical variable setting

This object can define a field to input an alphanumerical variable on the screen.

⇒ To add an object, choose the icon  and have a left click on the terminal screen.

⇒ When you click on the object while keeping the left button pressed, you move the alphanumerical variable following the mouse.

The properties of this object are:

☞ **Name** : It defines the object name, and must be unique.

☞ **Col** : It defines the first column of the screen where the text will be located.

☞ **Row** : It defines the line of the screen where the text will be located.

☞ **Variable** : It defines the alphanumerical variable to display and to affect the input value.

☞ **Help** : It indicates if a help menu is available with this input. If it is on Yes, it means that when the navigator ">" is located on this object, the Help led flashes and the associated help page can be displayed with the Help key.

☞ **Help Page** : It indicates the name of the page which will be as an help. To display it, the Help property must be true.

☞ **Password** : It indicates if an access code must be specified when modifying the field value (as soon as the Mod key is pressed).

☞ **Pass name** : It indicates what will be the access code of the codes list which will be useful.

☞ **Length** : It defines the maximum number of characters which can be input.

☞ **Navig** : It indicates the scheduling of the different input fields. The navigator ">" shows which field will be affected on the Mod key-pressed. To pass the indicator ">" from one field to the other, you just have to use the up and down arrows. To schedule the navigator, we indicate a number in the parameter "Navig". If this value is 1, the indicator will be at this place each time we arrive on this page. A down arrow key-pressed can pass to the field with the "Navig" parameter equal to 2, etc. To come back, you just have to press the up arrow key. All the "Navig" parameters on a page must be different. If 2 are the same, only one will receive the navigator ">" and be able to be modified. In the same way, the zero value in that field is not allowed.

Example :

Creation scheduling of the different inputs fields of a page

Input field n°1 : Navigator=2

Input field n°2 : Navigator=1

Input field n°3 : Navigator=3

Input field n°4 : Navigator=0

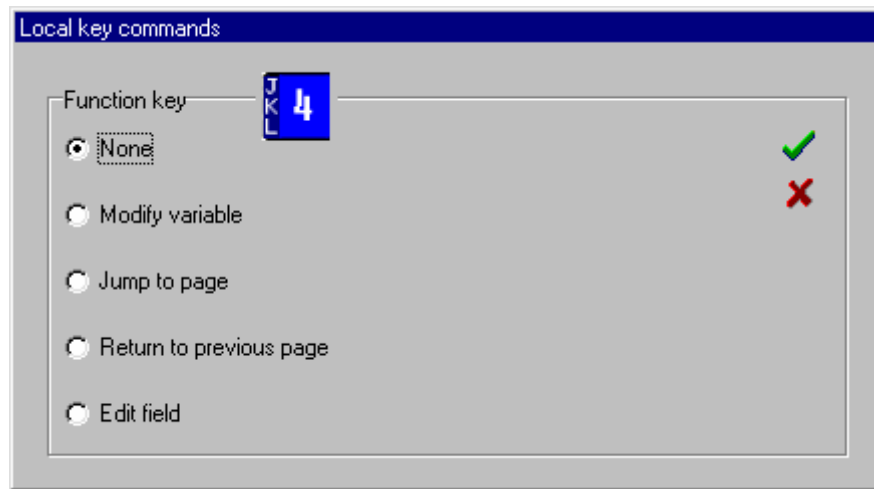
Input field n°5 : Navigator=3

Input field n°6 : Navigator=8

When the page is displayed, the indicator is on the input field number 2. A down arrow key-pressed changes it to the number 1, another one to the number 5. Actually, fields number 3 and 5 have the same property, but only the last input field will be addressed by the navigator. A new down arrow key-pressed goes to number 6. The up arrow key-pressed passes in the opposite way. The field number 4 will never have the indicator.

☞ **Confirm** : It indicates if a confirmation screen must appear after the input setting (ENTER key-pressed). The confirmation screen does not need to be written by the user. To confirm, you have to press the key 2, and the key 3 for cancelling.

5-4-8- Programming of the simple dynamical keys



The defined function for a key can be done only on the display of the current page. Their programming has priority regarding the global key functions programming. There are 4 associated functions available for a key:

☞ **Modify a variable** : It can modify a bit of a variable. For that, you have to specify the name of the variable and the bit number to change (bit 1 is least significant bit). The modification can be done by several types (parameter to choose):

- ⇒ direct : The bit is '1' when the key is ON and '0' when it is OFF.
- ⇒ flip-flop : A key pressed changes the bit from '0' to '1' or from '1' to '0'.
- ⇒ bit set : A key pressed changes the bit to '1'
- ⇒ bit reset : A key pressed changes the bit to '0'

☞ **Jump to page** : It displays another page. This new page must be specified and defined by the user. When the key is pressed, a confirmation page can be inserted. It is pre-defined and does not need to be made by the user.

☞ **Return to previous page** : When the key is pressed, the screen displays the last displayed screen.

☞ **Edit field** : When the key is pressed, you can input a field which must be specified by the user.

Example 1:

Page name: CurrentPage

```
Key 2 : Function : Modify a variable ;
        Type=direct ; Variable=Var ; Bit n°6
Key 3 : Function : Modify a variable ;
        Type=flip-flop ; Variable=Var ; Bit n°3
Key 4 : Function : Modify a variable ;
        Type=bit set ; Variable=Var ; Bit n°5
Key 5 : Function : Modify a variable ;
        Type=bit reset ; Variable=Var ; Bit n°5
```

During the display of CurrentPage, the following operations are giving those results:

- ⇒ Key 2 pressed : Var=XX1XXXXXb
- ⇒ Key 2 not pressed : Var=XX0XXXXXb
- ⇒ If Var=XXXXX0XXb and one key-pressed on 3 : Var=XXXXX1XXb
- ⇒ If Var=XXXXX1XXb and one key-pressed on 3 : Var=XXXXX0XXb

⇒ Key 4 pressed : Var=XXX1XXXXb
 ⇒ Key 5 pressed : Var=XXX0XXXXb

Example 2:

Global declared key :

Key 4 : Function : Return to previous page

Key 5 : Function : Return to previous page

Page name: CurrentPage

Key 2 : Function : Jump to page;

Page name: NextPage

Key 3 : Function : Return to previous page

Key 4 : Function : Edit field ;

Field name =CurrentPageField ;

During the display of CurrentPage, the following operations are giving those results:

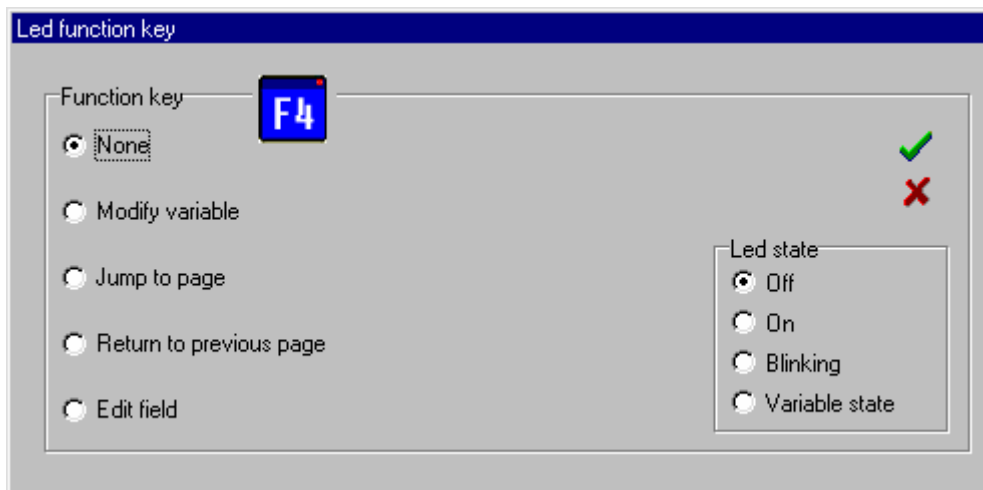
⇒ A press on key 2 displays NextPage. The declared functions for the dynamical keys of CurrentPage are not valid any more (only those declared in NextPage are valid).

⇒ A press on key 3 displays the last displayed page.

⇒ A press on key 4 edits the field CurrentPageField. This function has priority regarding the global defined function.

⇒ A press on key 5 displays the last displayed page.

5-4-9- Programmation of the led function keys



The defined function for a led and a key can only be done on the current page display. Their action has priority regarding to the programmation which can be made in a global way on this key. There are 4 available associated functions:

✚ **Modify a variable** : It can modify a bit of a variable. For that, you have to specify the name of the variable and the bit number to change (bit 1 is least significant bit). The modification can be done by several types (parameter to choose):

- ⇒ direct : The bit is '1' when the key is ON and '0' when it is OFF.
- ⇒ flip-flop : A key pressed changes the bit from '0' to '1' or from '1' to '0'.
- ⇒ bit set : A key pressed changes the bit to '1'
- ⇒ bit reset : A key pressed changes the bit to '0'

✚ **Jump to page** : It displays another page. This new page must be specified and defined by the user. When the key is pressed, a confirmation page can be inserted. It is pre-defined and does not need to be made by the user.

✚ **Return to previous page** : When the key is pressed, the screen displays the last displayed screen.

✚ **Edit field** : When the key is pressed, you can input a field which must be specified by the user.

The state of the led associated to the key can be defined by 4 ways:

↵ **Off** : the led will never be on

↵ **On** : the led will always be on

↵ **Blinking** : the led will flash

↵ **Variable** : the led will be same as the bit of the variable specified in the parameters of the function "Modify a variable".

Example 1:

Page name: CurrentPage

```
Key F1 : Function Key : Modify a variable ;
        Type=direct ; Variable=Var ; Bit n°6
        Function Led : Variable
Key F2 : Function : Modify a variable ;
        Type=direct ; Variable=Var ; Bit n°6
        Function led : variable
Key F3 : Function : Modify a variable ;
        Type=flip-flop ; Variable=Var ; Bit n°3
        Function led: variable
Key F4 : Function : Modify a variable ;
        Type=bit set ; Variable=Var ; Bit n°5
        Function led: variable
```

During the display of CurrentPage, the following operations are giving those results:

```
⇒ Key F1 ON : Var=XX1XXXXXb and led ON
⇒ Key F1 OFF : Var=XX0XXXXXb and led OFF
⇒ If Var=XXXXX0XXb and key F2 ON : Var=XXXXX1XXb and led ON
⇒ If Var=XXXXX1XXb and key F2 OFF : Var=XXXXX0XXb and led OFF
⇒ Key F3 ON : Var=XXX1XXXXb and led ON
⇒ Key F4 ON : Var=XXX0XXXXb and led OFF
```

Example 2:

Global declared key :

```
Key F3 : Function : Return to previous page
        Function led: Off
Key F4 : Function : Return to previous page
        Function led : blinking
Page name: CurrentPage
Key F1 : Function : Jump to page;
        Page name: NextPage
        Function led : Off
Key F3 : Function : Return to previous page
        Function led : On
Key F4 : Function : Edit field ;
        Field name =CurrentPageField ;
        Function led : Blinking
```

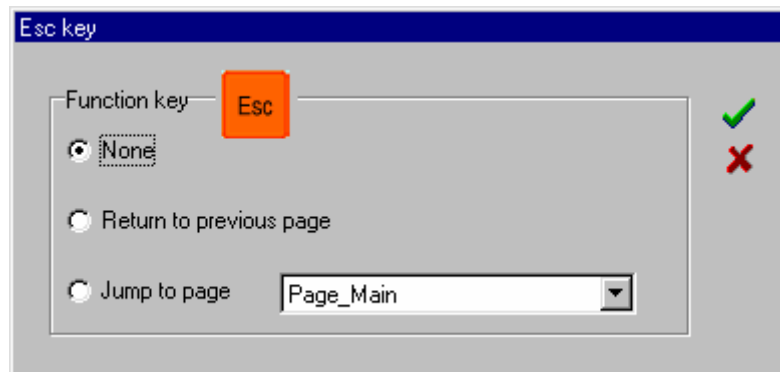
During the display of CurrentPage, the leds are like this:

```
⇒ led F1 : Off
⇒ led F2 : On
⇒ led F3 : Blinking
⇒ led F4 : Blinking
```

The following operations are giving those results:

```
⇒ A press on key F1 displays NextPage. The declared functions for the
dynamical keys of CurrentPage are not valid any more (only those declared in
NextPage are valid).
⇒ A press on key F2 displays the last displayed page.
⇒ A press on key F3 edits the field CurrentPageField. This function has
priority regarding the global defined function.
⇒ A press on key F4 displays the last displayed page.
```

5-4-10- Programming of the ESC key



The defined function for this key can only be executed for the current page. Its programming has priority regarding the programming which can be made in a global way on this key. The associated functions for this key can be:

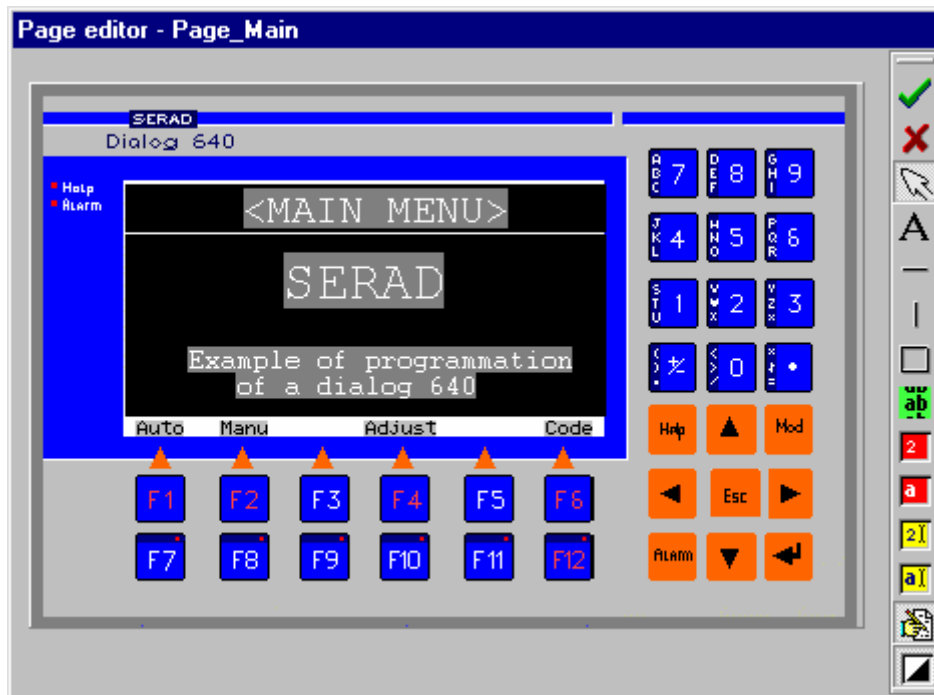
↳ **Return to previous page** : When the key is pressed, the screen displays the last displayed screen.

↳ **Jump to page** : It displays another page. This new page must be specified and defined by the user. When the key is pressed, a confirmation page can be inserted. It is pre-defined and does not need to be made by the user.

5-5- Dialog640 editor

5-5-1- Presentation of the editor

The editor is made of 2 windows.





The first window is the full screen editor. In this one, we can see the front page of the Dialog 640 and a tools bar on the right. On this screen, we can configure:


↳ the keys F1, F2, F3, F4, F5, F6 of the terminal with a left click on them


↳ the keys F7, F8, F9, F10, F11, F12 and their leds with a left click on the corresponding key.


↳ the ESC key with a left click on it


↳ the terminal screen


The tools bar is made of the icons: page edition validation , page edition cancelling . It is also made of these icons, to add objects on the terminal screen:


↳ Static text 


↳ Dynamic text 


↳ Numerical variable display 


↳ Alphanumerical variable display 


↳ Numerical variable setting 


↳ Alphanumerical variable setting 

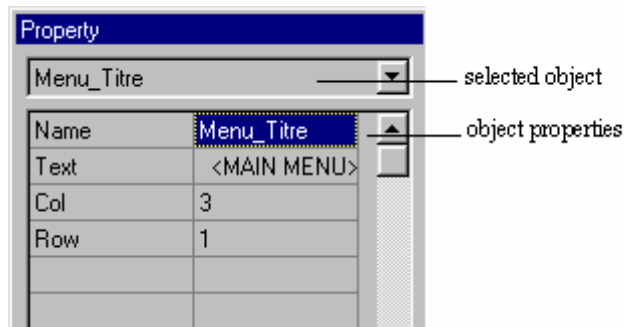
↳ Vertical line display 


↳ Horizontal line display 

↳ Rectangle display 

↳ Change the back screen colour 

The icon  can position on a screen object to modify its size or its properties. This icon is automatically as an object is added on the screen.



The last icon  can validate or not the second window which concerns the properties of the objects on the terminal screen. In this window, you can select the desired object defined with its name and display its properties. You can also select an object with a left click on this object on the terminal screen. The list of the properties on the left is different regarding the type of the selected object. The fields of the right column are the characteristics of the parameters of the left column, so you can modify them. Any modification of these parameters is shown by a modification of the object on the screen. In the same way, any modification of the object on the screen modifies its properties.

5-5-2- Fonts


For the Dialog 640 series, the character sizes may change. During the display of a page, the tools Static text, Dynamic text, Numerical value display, Alphanumerical variable display, Numerical variable setting, Alphanumerical variable setting have the same properties as the

Dialog 80 with one difference: it owns another property, which is the font used, from 1 to 8. From 1 to 4, the fonts are written in back on white background, and from 5 to 8, the fonts are written in white on black background. Then each number refers to a size of character:

- ↵ values 1 and 5 : size 3×4 mm, that is 16 lines of 40 characters
- ↵ values 2 and 6 : size 4×7 mm, that is 9 lines of 30 characters
- ↵ values 3 and 7 : size 5×8 mm, that is 8 lines of 26 characters
- ↵ values 4 and 8 : size 7×10 mm, that is 6 lines of 17 characters

5-5-3- Vertical line display

This kind of object permits to display a vertical line on the screen.


- ⇒ The adding of this object is made by selecting the icon  and by a left click on the terminal screen.
- ⇒ When you click on the object while keeping the left button pressed, you move the vertical line following the mouse.
- ⇒ When you click on the right or left edge of the object and keep the left button on, you can modify the height of the vertical line.

The properties of this object are:

- ↵ **Name** : It defines the object name, and must be unique.
- ↵ **X1** : It defines the X position of the vertical line, in pixels (from 1 to 240)
- ↵ **Y1** : It defines the high limit Y position of the vertical line in pixels (from 1 to 128).
- ↵ **Y2** : It defines the low limit Y position of the vertical line in pixels (from 1 to 128). Y2 must be higher than Y1.
- ↵ **Colour** : It defines the line colour(1: black, 0: white)

5-5-4- Horizontal line display

This kind of object permits to display a horizontal line on the screen.


- ⇒ The adding of this object is made by selecting the icon  and by a left click on the terminal screen.
- ⇒ When you click on the object while keeping the left button pressed, you move the horizontal line following the mouse.
- ⇒ When you click on the right or left edge of the object and keep the left button on, you can modify the length of the vertical line.

The properties of this object are:

- ↵ **Name** : It defines the object name, and must be unique.
- ↵ **X1** : It defines the left limit X position of the horizontal line, in pixels (from 1 to 240)
- ↵ **X2** : It defines the right limit X position of the horizontal line in pixels (from 1 to 240). X2 must be higher than X1.
- ↵ **Y1** : It defines the Y position of the horizontal line in pixels (from 1 to 128).
- ↵ **Colour** : It defines the line colour(1: black, 0: white)

5-5-5- Rectangle display


This kind of object permits to display a rectangle on the screen.

- ⇒ The adding of this object is made by selecting the icon  and by a left click on the terminal screen.
- ⇒ When you click on the object while keeping the left button pressed, you move the rectangle following the mouse.
- ⇒ When you click on the up or down edges of the object and keep the left button on, you can modify the height of the rectangle.
- ⇒ When you click on the right or left edges of the object and keep the left button on, you can modify the width of the rectangle.
- ⇒ When you click on a rectangle's corner and keep the left button on, you can modify both width and height of the rectangle.

The properties of this object are:

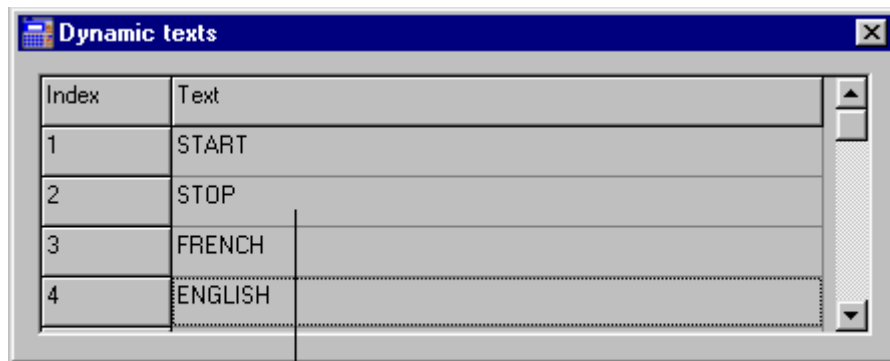
- ↻ **Name** : It defines the object name, and must be unique.
- ↻ **X1** : It defines the X position of the left corner, in pixels (from 1 to 240)
- ↻ **X2** : It defines the X position of the right corner, in pixels (from 1 to 240)
- ↻ **Y1** : It defines the Y position of the high corner in pixels (from 1 to 128).
- ↻ **Y2** : It defines the Y position of the down corner in pixels (from 1 to 128).
- ↻ **Line** : It defines the colour of the line of the rectangle (1: black, 0: white)
- ↻ **Fill** : It defines the filling colour of the rectangle (1: black, 0: white)

5-5-6- Change the back screen colour

Pressing this icon  changes the back screen colour, which is white when active and black when disable. It is better to make screens with a predominant black colour, to increase life of the panel.

6- DYNAMIC TEXTS

6-1- General presentation



Text associated with an index number

In this window we define the texts which will be displayed by a dynamic text object. The maximum number is 1.500 for a dialog 640 and 3.000 for a Dialog 80.

For example, let us imagine that you have a dynamic text object which must display the state of the machine. The coding property is binary type. There are 3 parameters to specify:

- ⇒ the number (Number) of possible texts to display
- ⇒ the starting address in the list (Index)
- ⇒ the text relative address (Variable)

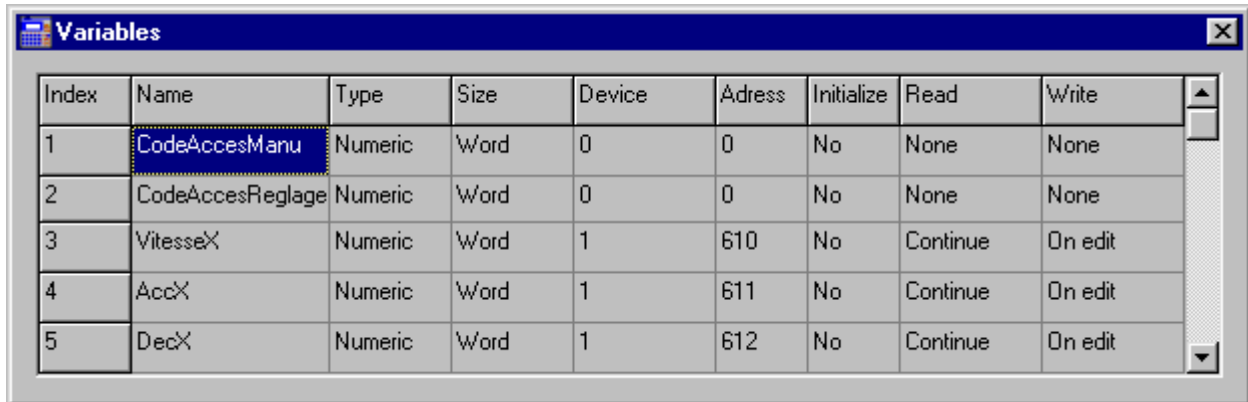
The texts to display are only two (Run and Alarm), the property 'Number' must be set to 2. If now the messages are in positions 1 and 2 in the list, it means that the starting address must be 1. You just have to create a variable, type Word for example, in the variable list. This variable will be similar as the machine state, that is '0' for run and '1' for alarm.

If the messages were in position 45 and 46, 'Number' would always be 2. However, the starting address 'Index' would be 45, and the variable value must be 0 or 1. Warning, you can not set the index to one and affect 44 or 45 to the variable, because 'Number' limits the variable from 0 to 1.


If the texts number were 45 ('Number'=45), the display would be allowed for variable includes from 0 to 44.

7- VARIABLES LIST

7-1- General presentation



Index	Name	Type	Size	Device	Adress	Initialize	Read	Write
1	CodeAccesManu	Numeric	Word	0	0	No	None	None
2	CodeAccesReglage	Numeric	Word	0	0	No	None	None
3	VitesseX	Numeric	Word	1	610	No	Continue	On edit
4	AccX	Numeric	Word	1	611	No	Continue	On edit
5	DecX	Numeric	Word	1	612	No	Continue	On edit

This window shows the variable list used by a project. It is active or not if you press the icon . From this window, you can declare, modify or delete a variable and its properties. The number of declared variables is at most 5.000 for the Dialog 80 and 640.

7-2- Declaration and modification of a variable

To declare a new variable, the user must edit an empty wording (double left-click on the box), input a name for its variable and validate with the ENTER key. When a variable is declared, all its parameters have a default value.

↻ **Size** : Define the type used by the variable. It may be a word (signed-16 bits) or a DWord (signed-32 bits).

↻ **Type** : It is the representation of the variable in a numerical form (value) or an alphanumerical one (string of characters). A character is one byte. So an alphanumerical value can store 2 characters for a Word, and 4 ones for a DWord. If you need a bigger size variable, you must let as many free fields as the required size needs to. Those free fields must be just after the variable declaration, and must have the same size as the first one (even if it is not the same type). In the same way, the size of an alphanumerical variable is defined regarding the maximum number of characters which will be displayed or input by the objects defined in the different pages. For example, if you want to display (or input) at most 10 characters, the associated alphanumerical variable must have at least 10 bytes, that means 5 places if you are working in Word size, and 3 if you use Dwords. If the variable is defined as a Word located in address 26, the addresses 27, 28, 29, 30 must be free. If it is defined as a Dword located in the same address 26, only the addresses 27 and 28 must be free. However, it is better to define the free fields with a name like the variable name and an index (0, 1...). In the same way, it is better to put the alphanumerical values in a particular place of the 5.000 available places, to separate them of the numerical variables. It will then be easier to let free places if you want to modify the size of a variable.

↻ **Device** : This property indicates the peripheral number linked to the terminal.

↻ **Address** : It indicates the memory address of the peripheral where the variable can be read and / or written. Warning: this address must be accessible on the peripheral.

↻ **Initialize** : It sends the variable to the concerned peripheral when you switch on the terminal. This property permits to the terminal to be as a safe memory for the peripheral.

↳ **Read** : The terminal can read for many ways a variable of a peripheral:

⇒ in a continuous way : the variable is read while the terminal is in run. For a dynamic text, it is the index variable which is read.

⇒ on show : the variable is read each time that a page will be displayed and that it needs to display the variable. For a dynamic text, it is the index variable which is read.

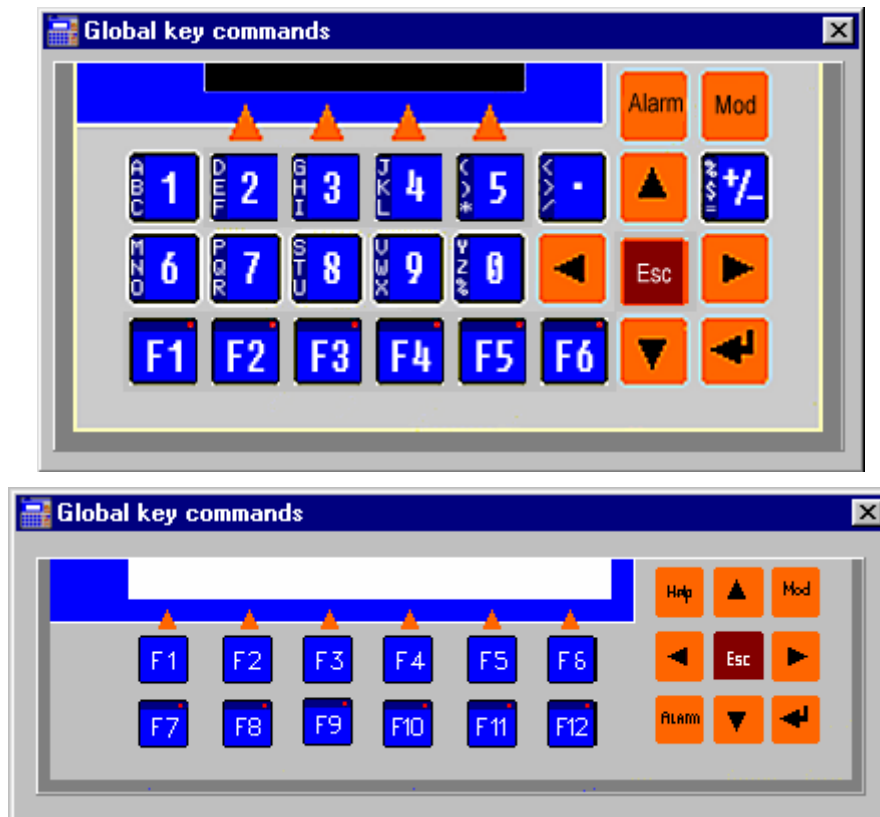
↳ **Write** : The terminal send the state of the variable to the peripheral when a variable input is done.


7-3- Deletion of a variable

The deletion of a variable is made by supressing all the characters which are making its name and validate the box with ENTER. The validation results in the deletion of all the next fields.

8- GLOBAL KEYS COMMANDS

8-1- General presentation



To access to one of these windows, you must use the icon . From this window, we can define a particular function which will be affected to a key. This function will be global. It means that it will be active at any time, except if another local definition is programmed on the current active page. A local programmed key will have priority regarding a global one.

The keys for which you can define a function are:

⇒ 2, 3, 4, 5, F1, F2, F3, F4, F5, F6 for the dialog80

⇒ F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12 for the dialog640

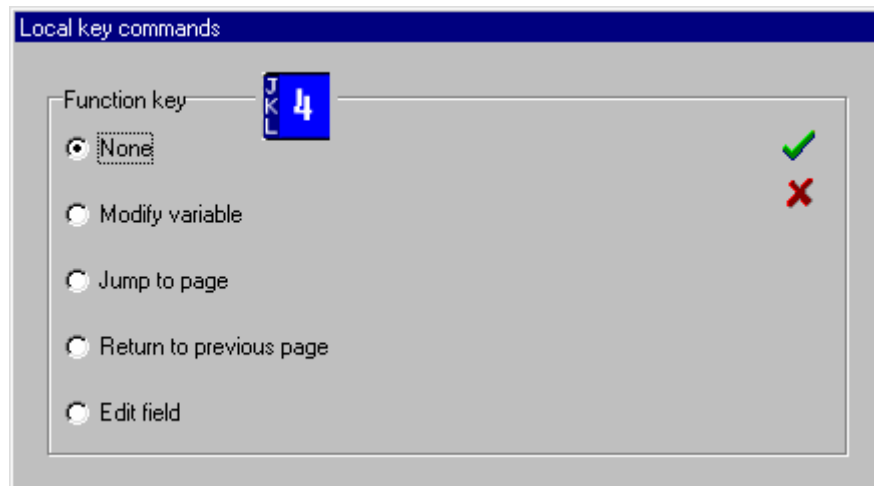
On each terminal, we can also program the ESC key. We can also drive the leds associated with some keys. To define a key function, you have to click on the key with the left button.

8-2- Programmation of the dynamic keys

The keys for which we can associate a function are:

⇒ 2, 3, 4, 5 for the dialog80

⇒ F1, F2, F3, F4, F5, F6 for the dialog640



The associated functions available are:

↳ **Modify a variable** : It can modify a bit of a variable. For that, you have to specify the name of the variable and the bit number to change (bit 1 is least significant bit). The modification can be done by several types (parameter to choose):

- ⇒ direct : The bit is '1' when the key is ON and '0' when it is OFF.
- ⇒ flip-flop : A key pressed changes the bit from '0' to '1' or from '1' to '0'.
- ⇒ bit set : A key pressed changes the bit to '1'
- ⇒ bit reset : A key pressed changes the bit to '0'

↳ **Jump to page** : It displays another page. This new page must be specified and defined by the user. When the key is pressed, a confirmation page can be inserted. It is pre-defined and does not need to be made by the user.

↳ **Return to previous page** : When the key is pressed, the screen displays the last displayed screen.

Example 1:

```
Key 2 : Function : Modify a variable ;
        Type=direct ; Variable=Var ; Bit n°6
Key 3 : Function : Modify a variable ;
        Type=flip-flop ; Variable=Var ; Bit n°3
Key 4 : Function : Modify a variable ;
        Type=bit set ; Variable=Var ; Bit n°5
Key 5 : Function : Modify a variable ;
        Type=bit reset ; Variable=Var ; Bit n°5
```

During the terminal run and if no local instructions for the keys 2, 3, 4, 5 are declared, the following operations are giving those results:

```
⇒ Key 2 pressed : Var=XX1XXXXXb
⇒ Key 2 not pressed : Var=XX0XXXXXb
⇒ If Var=XXXXX0XXb and one key-pressed on 3 : Var=XXXXX1XXb
⇒ If Var=XXXXX1XXb and one key-pressed on 3 : Var=XXXXX0XXb
⇒ Key 4 pressed : Var=XXX1XXXXb
⇒ Key 5 pressed : Var=XXX0XXXXb
```

Example 2:

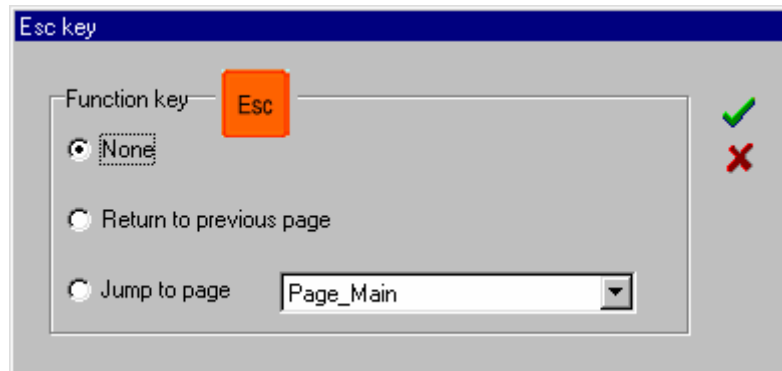
```
Global declared key :
Key 2 : Function : Jump to page
        Page name: NextPage
Key 3 : Function : Return to previous page
Key 4 : Function : Jump to page
        Page name: NextPage
Page name: CurrentPage
Key 2 : Function : Jump to page;
        Page name: NextPage
Key 4 : Function : Return to previous page
```


Key 5 : Function : Return to previous page

During the display of power-on, the following operations are giving those results:

- ⇒ A press on key 2 displays NextPage.
- ⇒ A press on key 3 displays the last displayed page.
- ⇒ A press on key 4 jumps to NextPage, except if CurrentPage is displayed. In this case, it displays the last displayed page.
- ⇒ A press on key 5 displays the last displayed page when CurrentPage is on.

8-3- Programming of the ESC key



The associated functions are:

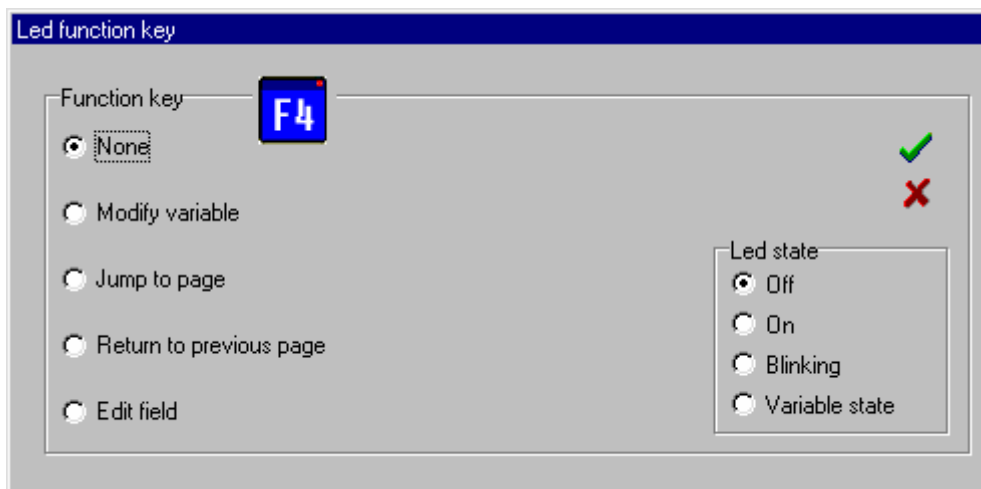
↵ **Return to previous page** : When the key is pressed, the screen displays the last displayed screen.

↵ **Jump to page** : It displays another page. This new page must be specified and defined by the user. When the key is pressed, a confirmation page can be inserted. It is pre-defined and does not need to be made by the user.

8-4- Programming the LED function keys

The keys for which we can associate a function are:

- ⇒ F1, F2, F3, F4, F5, F6 for the dialog80
- ⇒ F7, F8, F9, F10, F11, F12 for the dialog640



The associated functions are:

↵ **Modify a variable** : It can modify a bit of a variable. For that, you have to specify the name of the variable and the bit number to change (bit 1 is least significant bit). The modification can be done by several types (parameter to choose):

- ⇒ direct : The bit is '1' when the key is ON and '0' when it is OFF.
- ⇒ flip-flop : A key pressed changes the bit from '0' to '1' or from '1' to '0'.
- ⇒ bit set : A key pressed changes the bit to '1'
- ⇒ bit reset : A key pressed changes the bit to '0'

↵ **Jump to page** : It displays another page. This new page must be specified and defined by the user. When the key is pressed, a confirmation page can be inserted. It is pre-defined and does not need to be made by the user.

↵ **Return to previous page** : When the key is pressed, the screen displays the last displayed screen.

The state of the led associated to the key can be defined by 4 ways:

↵ **Off** : the led will never be on

↵ **On** : the led will always be on

↵ **Blinking** : the led will flash

↵ **Variable** : the led will be same as the bit of the variable specified in the parameters of the function "Modify a variable".

Example 1:

```
Key F1 : Function Key : Modify a variable ;
        Type=direct ; Variable=Var ; Bit n°6
        Function Led : Variable
Key F2 : Function : Modify a variable ;
        Type=flip-flop ; Variable=Var ; Bit n°3
        Function led : variable
Key F3 : Function : Modify a variable ;
        Type=bit reset ; Variable=Var ; Bit n°5
        Function led: variable
Key F4 : Function : Modify a variable ;
        Type=bit reset ; Variable=Var ; Bit n°5
        Function led: variable
```

When the system is switched on, the following operations are giving those results:

```
⇒ Key F1 on : Var=XX1XXXXXb and led ON
⇒ Key F1 off : Var=XX0XXXXXb and led OFF
⇒ If Var=XXXXX0XXb and F2 ON : Var=XXXXX1XXb and led ON
⇒ If Var=XXXXX1XXb and F2 OFF : Var=XXXXX0XXb et led OFF
⇒ Key F3 ON : Var=XXX1XXXXb and led ON
⇒ Key F4 ON : Var=XXX0XXXXb and led OFF
```

Example 2:

```
Key declared in local for CurrentPage
Key F3 : Function key : Return to previous page
        Function led : Off
Key F4 : Function Key : Return to previous page
        Function led : Flashing

Global declared keys :
Key F1 : Function : Jump to page ;
        Page name : NextPage ;
        Function led : Off ;
Key F2 : Function : Return to previous page;
        Function led : On
Key F3 : Function : Jump to page ;
        Page name : NextPage;
        Function led : Flashing ;
```

During the power-on, the leds are like this:

```
⇒ led F1 : Off
⇒ led F2 : On
```

Dialog80 and Dialog640 operator terminal documentation

⇒ led F3 : Off on CurrentPage, flashing on the others

⇒ led F4 : Flashing on CurrentPage

The following operations are giving those results:

⇒ A press on F1 displays NextPage. The declared function keys for CurrentPage are not active any more, only those declared for NextPage are.

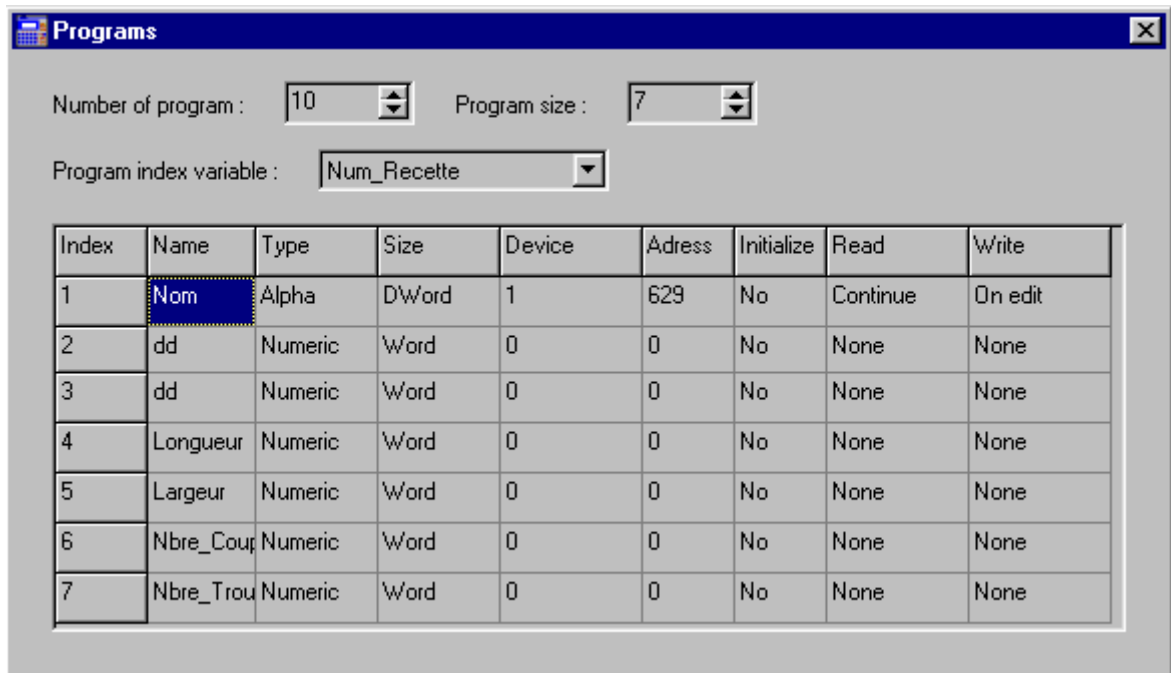
⇒ A press on F2 displays the last displayed page.

⇒ A press on F3 jumps to NextPage, except if CurrentPage is displayed. In this case, it returns to the last displayed page.

⇒ A press on F4 returns to the last displayed page.

9- PROGRAMS

9-1- General presentation



The word program defines a parameters group for a product. For example for cutting a piece, this one can be described with a name, cutting lengths, etc. This group of parameters is a program. A program is then defined with its size which is the number N of variables. For a system one program is not enough. We define as many programs as the number M of different products, and then M is the number of program.

To go from one program to the other, we use an index variable, which state defines the selected program. If you want to display the value of a program variable like a product length with a field 'display of a numerical value', we define the variable property of this field with the variable 'program index variable'. The terminal will display the product length regarding the index variable. If this one is 1, the length will be the one of the program 1. If it is 5, the length will be the one of the program 5. To go from one program to the other, you must modify the value of the index variable. Then you must prepare a field to modify this value to change the selected programs.

For the Dialog 80 and Dialog 160, we can have up to 10.000 program variables. It means that the multiplication M.N must be inferior to 10.000. N is the number of variable in one program, whatever its type is (a word is considered as big as a Dword). M is the number of programs. For example, if a product is defined with 250 variables, we will not be able to store more than 40 programs. In this screen, we define the size and the number of the programs, and the associated variables. Each variable value will be set from the terminal.

9-2- Declaration of a program

To define a program, the first thing is to define its size. It can be modified at any time and it specifies the number of lines to declare the program variables. Then we define the program variables. The variables characteristics for a program (type, size, and declaration of a characters string) are similar to the variable characteristics itself (see variables list). Then you can define

the program number and the index variable. The index variable is one of those declared in the variables list.

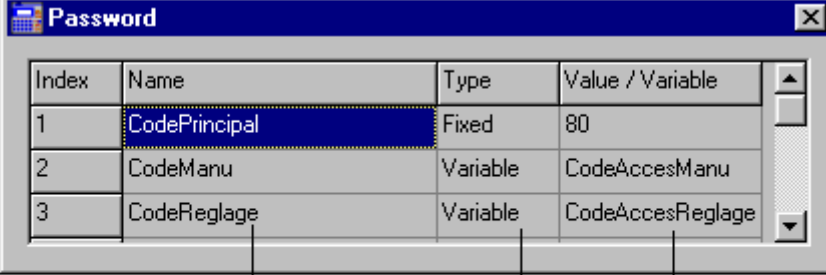
9-3- Deletion of a program

To delete a program variable, you just have to suppress the wording of the selected program. When this is done, all the associated fields are empty.

To totally delete a program you must suppress every variable and set to 1 the fields: size and number of program.

10- PASSWORD

10-1- General presentation



The screenshot shows a window titled "Password" with a table containing three rows. The first row is highlighted. Below the table, there are three columns with labels: "Name of the code", "Fixed or variable type", and "Value for a fixed type / Variable name for a variable type".

Index	Name	Type	Value / Variable
1	CodePrincipal	Fixed	80
2	CodeManu	Variable	CodeAccesManu
3	CodeReglage	Variable	CodeAccesReglage

Name of the code Fixed or variable type Value for a fixed type
Variable name for a
variable type

The list of passwords can define the type of each one. You can have 2 different types:

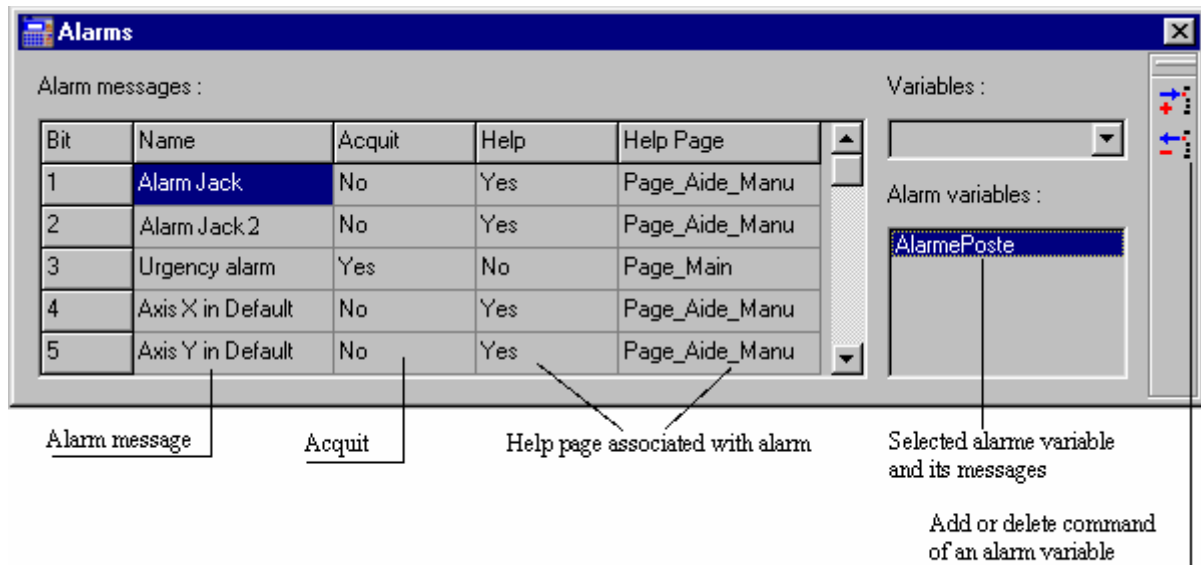
↳ **fixed** : In this case, the value must be specified with 4 figures at most.

↳ **variable** : In this case, we can specify a numerical value (a variable of Word size is enough). Its content will be the code value.

The access codes may be defined to link some pages, or to input a numerical or alphanumerical value. Their property need to specify one of the variables in the passwords list. The access code page is pre-defined, it does not need to be made by the user: when the user is loading its password, the characters displayed are *. The codes are numerical values. It is better not to use codes beginning with 0 as 0XXX, 00XXX, 000X and 0000. In this case, the codes XXX, XX, X, empty and 0 will be true.

11- ALARMS


11-1- General presentation




This window gathers all the alarm variables declared for the system. Each bit of a variable is corresponding to an alarm (bit=0, no alarm, bit=1, alarm on). When the system detects an alarm, the “alarm” led is flashing. As soon as this led is flashing, the user can see the element(s) which started the alarm, with pressing the ALARM key. The alarm page which there appears is a pre-defined page, telling the names of the different active alarms. On this page, we can also see an arrow in the left-upper corner. It shows which is the selected alarm and permits to know the associated characteristics for this alarm. To select and know the characteristics of another alarm, the user can serve as the up and down arrows. The alarm characteristics are: the help page and the acquittal mode. The help page is a pre-defined by the user page which can be reached when the associated alarm is present and is selected in the alarm page. You can see that there is an help page by the flashing of the HELP led. Then you just have to press the Help key, and then go to it. The acquittal mode can be defined without acquit (select ‘no’), it means that as soon as the default disappears, the alarm is not shown any more. If you select ‘yes’, the alarm signal is still present until the alarm is off and the user acquits it. The acquittal is made in the ‘Alarm’ page, by pressing the ‘2’ key for a Dialog 80, and the ‘F1’ key for a Dialog 640.

11-2- Adding and configuring an alarm

To add an alarm, you have to select a variable representative of the alarms in the variables list.

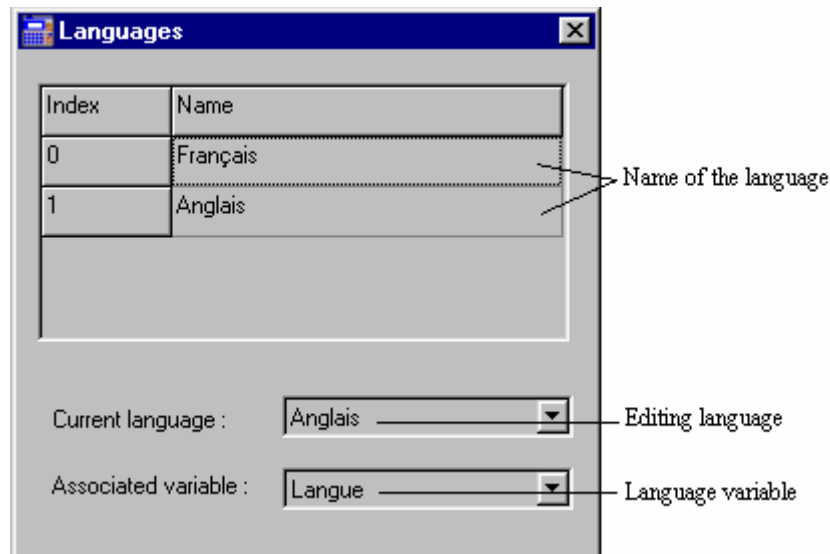
Then with the icon , you add it into the alarm variables list. You select it in this list. If it has not been configured, the messages list is empty. To configure it, you define the wordings corresponding to the alarm messages. Each wording shows a bit of the alarm variable. If this variable is a Word, we will have 16 different messages, 32 if it is a DWord. For each alarm message, we can define its acquit mode and the associated help page. To define an alarm message, you must first define the wording, and then its parameters (Acquit, Help...).

11-3- Cancelling an alarm

To suppress an alarm variable from the list, you just have to select it and activate the icon . To suppress the alarm message associated to an alarm variable, you just have to suppress the wording and all the associated configuration will be erased.

12- LANGUAGES

12-1- General presentation



The number of languages which can be programmed is depending of the defined choice made during the declaration of the project. It can be from 1 to 4. To define the language name, you just have to double-click on one of the wordings and specify it. The default current language is the one with the number 0. The associated variable tells to the terminal which language must be displayed: if its value is 0, it will be the language number 0, if it is 1, language number 1, etc. This variable can be changed on the terminal at any times, and so change the language. To define a language's texts, you must first define all the texts in the default language. Then, from this languages window, you modify the current language. All the texts are not modified or cancelled, but are still in the default language. You just have to translate them in the current language. After saving the project, you can change from one language to the other one, and then you can see that texts appear on the different pages in the selected language.

13- PROTOCOLS

13-1- Definition of the protocol

To declare the communication protocol used by the terminal Dialog 80 or Dialog 640, you have to select the command “protocol” of the Options menu. The selection opens a dialog box which contains the different parameters. The terminals Dialog 80 and Dialog 640 are able to manage the following protocols:

↳ MODBUS RTU MAITRE

↳ CANOPEN

13-2- MASTER RTU MODBUS

13-2-1- Declaration of the ModBus

In this window, we define the communication protocol for the ModBus. This choice needs a parameters setting for some communication characteristics. Among those characteristics, there are the serial channel configuration with its type, its speed, the data bits number, the parity, the stop number, the time-out and the number of allowed retries. You also have to define the state table and the command table if they are used by the peripheral. Moreover, you have to specify the number of the peripheral and the start address of these tables in the peripheral. When it is declared and the project is sent to the terminal, the state table is sent continuously to the peripheral and the command table is read continuously.

Warning: The peripheral addresses defined for the different elements of the table must be read and/or write addresses.

13-2-2- Presentation of the state table

The state table ids sent continuously in the peripheral. It is made like this:

	15	0
Word 0	Keys 1	Keys 0
Word 1	Keys 3	Keys 2
Word 2	Keys 4	Last key pressed
Word 3	Number of the running page	
Word 4	Setting state	Number of the last setting
Word 5	Led ON	Led OFF
Word 6	N° of the OS version	
Word 7	Counter of exchange	

↳ Each bit of the bytes “key 0” to “key 4” corresponds to the state of a key. When one of this bit is set, the attached key is pressed. In the other case, the key is slack. Because of the lower number of keys on the Dialog 80 , for this terminal the byte keys 4 is not used. The bits are corresponding to keys following this:

Keys[0]								
Bits	0	1	2	3	4	5	6	7
Dialog 80	F1	F2	F3	Enter	Down	F6	F5	F4
Dialog 640	F7	F8	F12	None	None	F11	F10	F9
Keys[1]								
Bits	0	1	2	3	4	5	6	7
Dialog 80	6	7	8	Right	ESC	Left	0	9
Dialog 640	F1	F2	F6	None	None	F5	F4	F3
Keys[2]								
Bits	0	1	2	3	4	5	6	7
Dialog 80	1	2	4	+/-	Up	.	5	4
Dialog 640	None	9	Enter	Right	mod	.	3	6
Keys[3]								
Bits	0	1	2	3	4	5	6	7
Dialog 80	None	None	None	mod	Alarm	Help	Shift	None
Dialog 640	None	8	Down	Esc	Up	0	2	5
Keys[4]								
Bits	0	1	2	3	4	5	6	7
Dialog 80	None	None	None	None	None	None	None	None
Dialog 640	None	7	Alarm	Left	Help	+/-	1	4

↳ The byte last key pressed gives the value of the key which has been pressed last. The connection between the keys and the decimal values is:

- ⇒ keys F1 to F12 : from 97 to 108 (61h to 6Ch)
- ⇒ keys Right, Left, Up, Down : from 28 to 31 (1Ch to 1Fh)
- ⇒ key ESC : 27 (1Bh)

- ⇒ key MOD : 77 (4Dh)
- ⇒ key ALARM : 76 (4Ch)
- ⇒ key : 69 (45h)
- ⇒ key POINT : 46 (2Eh)
- ⇒ key RETURN : 13 (0Dh)
- ⇒ key SHIFT : 72 (48h)
- ⇒ key SIGN : 83 (53h)
- ⇒ keys 0 to 9 : from 48 to 57 (30h to 39h)

↵ Number of the current page sends the index of the current page displayed on the terminal. This index is similar to the one in the page list.

↵ Number last input indicates the capture field number which has been activated last. The returned number corresponds to the scheduling number of creation of this field.

↵ Capture state sends the state of the last capture. The bit 8 corresponds to a capture made correctly and validate with RETURN. The bit 9 corresponds to a capture running. The other bits are not used.

↵ The bytes Led ON and Led OFF show the leds state on the Dialog 80 and Dialog 640.

Led F1 to F6 for the dialog80 : bit 0 to bit 5 and bit 8 to bit 13

Led F7 to F12 for the dialog640 : bit 0 to bit 5 and bit 8 to bit 13

Led HELP : bit 6 and bit 14

Led Alarm : bit 7 and bit 15

For a combination of the 2 bits of the registers Led ON and Led OFF corresponds a state of the led:

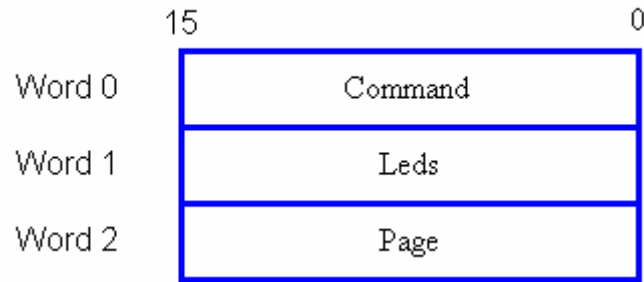
- ⇒ Led ON and Led OFF = 0 : led off
- ⇒ Led ON = 1 and Led OFF = 0 : led flashing
- ⇒ Led ON and Led OFF = 1 : led on

↵ The register version number gives the version of the operating system of the terminal on a BCD form (For example 0100 for a version 1.0)

↵ The register exchanges counter is incremented for each exchange between the peripheral and the terminal. It can be useful to detect communication failures.

13-2-3- Presentation of the command table

The command table is made of 3 consecutive words stored in the peripheral. They are continuously read by the terminal. Referring to the state of these words, the terminal does commands told by the peripheral: forcing leds state, displaying a page...



↳ The word 0 can activate or not each of the available commands. The bit 0, when set, can force the state of the leds contained in the word 1. The bit 1, when set, displays the page which number is in the word 2. The bit 2, when set, can inhibit the keys. The bit 3, when set, forces the terminal buzzer.

↳ The word 1 is reserved to define the leds state. This word is made of 2 bytes Led ON (bit8 to bit15) and Led OFF (bit0 to bit7), which any of the bits can define a state for any led. The bits attribution for the leds is as follows:

Led F1 to F6 for the dialog80 : bit 0 to bit 5 and bit 8 to bit 13

Led F7 to F12 for the dialog640 : bit 0 to bit 5 and bit 8 to bit 13

Led HELP : bit 6 and bit 14

Led ALARM : bit 7 and bit 15

For a combination of the bits of the bytes Led ON and Led OFF, is corresponding a leds state:

⇒ Led ON and Led OFF = 0 :Led off

⇒ Led ON = 1 and Led OFF = 0 :Led flashing

⇒ Led ON and Led OFF = 1 :Led on

13-3- CANopen

13-3-1- Introduction

The CAN bus (Controller Area Network) appeared in the middle of the 80ies as an answer for the data transmission in the automotive fields. This kind of bus can have transmission speeds up to 1 Mb/s.

The CAN specifications are defining 3 layers among the ISO/OSI model: the physical one, the data linking one and the application one. The physical layer defines the data transmission mode regarding the transmission support. The data linking layer is the nucleus of the CAN protocol because it deals with the frame to send, the arbitration, the defaults detection, etc. The last layer is also called CAL (CAN Application Layer). It is a general description of the language for the CAN networks which offers many communication services.

CANopen is a type of network based on the serial bus system and the application layer CAL. CANopen offers only part of the communication services that CAL has at its disposal. Those are the necessary advantages that need small performances computer, without storage capability.

So the CANopen is an application layer standardised by the CIA (CAN In Automation) specifications: DS-201...DS-207

The network manager permits an easier network initialisation. The network can be extended with all the components the user wants to.

The CAN bus is a multi-master bus. The sent messages are identified, instead of the connected modules as in the other field-buses. The network elements are allowed to send their message each time the bus is free. Bus conflicts are solved with a priority level given to messages. The CAN bus emits messages divided among 2032 priority levels. All the network elements have the same rights, so this communication is possible only without master bus.

Each element is deciding itself when it has data to send. However it is possible to send data with another element. This demand is made with the distant frame.

The CANopen specifications (DS-201...DS-207) define the technical and functional characteristics needed by any device to be plugged in the network. The CANopen bus makes a distinction from the server devices and the client devices.

13-3-2- The CANopen communication

The CANopen communication profile permits to specify information for data exchange in real time and parameters. The CANopen uses optimised services following the data types:

↳ PDO (Process Data Object)

- ⇒ Data exchange in real time
- ⇒ High priority identifier
- ⇒ Synchronous or asynchronous transmission
- ⇒ 8 bytes (one message) maximum
- ⇒ Pre-defined format

↳ SDO (Service Data Object)

- ⇒ Access to the objects dictionary of a device
- ⇒ Low priority identifier
- ⇒ Asynchronous transmission
- ⇒ Data distributed in many telegrams
- ⇒ Data addressed with an index

The characteristics diffused on the CAN bus are received and evaluated by all the connected devices. Each service of a CAN device is configured by a COBID (Communication Object Identifier). The COBID is an identifier which characterises the message. It tells to a device if the message must be taken in account. For each service (PDO or SDO), it is necessary to specify a COBID during the emission (sending a message) and a reception COBID (receiving a message). For the first SDO server, the COBID is fix and can not be modified remotely. Moreover, it is calculated from the NodeID. The NodeID is the parameter which characterises the device and permits the unique access to it.

PDO (Process Data Object)

It is a data exchange arbitrated between 2 modules. The PDO can transfer in turn some synchronisation or controlled events to realise the message sending request. With the controlled events mode, the load of the bus can be very reduced. A device can therefore realise a high performance with a low transfer rate.

The data exchange with the PDO uses the CAN advantages:

- ↳ Sending messages can be done from an asynchronous event (controlled event)
- ↳ Sending messages can be done from the reception of a synchronisation event.
- ↳ Recovery from a remote frame.

SDO (Service Data Object)

It is a data exchange point to point. A device is asking for an access in the objects list of a SDO. This one sends back an information corresponding to the type of request made by the caller. Each SDO can be either client and / or server. A server SDO can not send a request to another SDO, but it can answer any request from another client SDO. Unlike the PDOs, the SDOs must follow a particular communication protocol . The frame to send must have 8 bytes :

- ↳ Domain Protocol (Byte 0) : it defines the command (Upload, Download,...)
- ↳ Index on 16 bits (Bytes 1 et 2) : It defines the objects dictionary address.
- ↳ Sub-index on 8 bits (Octet 3) : It defines the element of the selected object in the dictionary
- ↳ Parameter (Octet 4 à 7) : It defines the value of the parameter read or written.

The network manager has a simplified mode to start the network up. The network configuration is not necessary in all the cases. The default configuration of the parameters may be enough. If the user wants to optimise the CANopen network or increase its functionalities, he can the modify himself these parameters. In the CANopen networks, each device has the same rights and the data exchange is directly regulated between each participant device.

The profile of a device defines the necessary parameters for a communication. The contents of this profile is specified by the constructor. Devices with the same profile are directly interchangeable. Most of the parameters are described by the constructor. The profile has empty places too which are for the future functionality extensions.

In most of the master/slave buses, the efficiency of the master determinates the comportment of the whole network. Moreover, slaves can not communicate directly one with the other. All these characteristics are increasing the transmission errors. CANopen suppress all of these drawbacks. The timing comportment can be specified individually for each respective task of the participant devices. Like that, the whole communication system does not need to have more efficiency if only some of the devices need so. Moreover, an automatic task can be separated for each of the participant devices. So the performances of the network manager can be used in an optimised way and can increase at any time by adding new participant devices.

The variables mapping used during the PDO type exchanges permits to use in an optimal way the current bandwidth of the bus. CANopen determinates default values of all the parameters.

13-3-3- Characteristics

The DWIN software offers the following possibilities to program exchanges with the CANopen protocol:

- ↳ A default SDO server.
- ↳ A SDO client to access to variables such as PLCs and PC boards.
- ↳ An peripheral affectation table to associate each terminal variable with a place in any CANopen device dictionary.

13-3-4- Dictionary

The dictionary contains the different parameters and variables of the terminal CANopen board. These parameters have no use for the user and are then not showed. However, it is necessary to find the important devices information to exchange data. You can find here part of the MCS32 EX dictionary, including variables, their index and their sub-index.

Index	Sub-idx	Name	Type	Attr.	Default
7180	1 to FEh	Read 32 bits variables	signed32	ro	none
7200	1 to FEh	Read 8 bits variables	unsigned8	ro	none
7280	1 to FEh	Read 16 bits variables	unsigned16	ro	none
8180	1 to FEh	Write 32 bits variables	signed32	wo	none
8200	1 to FEh	Write 8 bits variables	unsigned8	wo	none
8280	1 to FEh	Write 16 bits variables	unsigned16	wo	none

Those are the important information to exchange data with this device. You have to know too parameters located in the indexes 1200h, 1201h, etc. These parameters are the server COBID of the device. Parameters of the index 1200h are fixed by the NodeID. This value is located in the index 100Bh and can be fixed on the device by switches or software. For any other device, it is necessary to have the dictionary and know the different mentioned indexes. Moreover, you have to master what are the device characteristics (variables, inputs, outputs indexes, etc.).

13-3-5- Configuration

The operator terminal is not configurable remotely from another CANopen peripheral. It is necessary to configure it with the configuration CANopen window. This one is defined in 3 parts: the protocol configuration, the peripheral table one, and the states and commands table.

The screenshot shows the 'Protocol' configuration window. On the left, there are labels for various settings: 'Protocol choice' (CANopen), 'Number of the terminal Node-ID' (Node-ID: 34), 'CAN bus speed' (Speed: 500 KHz), 'Delay before a retry' (Timeout: 50, 1/100 s), 'Number of retry' (Retry: 10), 'Wait for a NMT signal before running' (Wait for NMT Start: unchecked), 'Send a NMT start to all the peripheral' (Send NMT Start: checked), and 'Peripheral identification by Node-ID' (Default SDO: checked). On the right, there are sections for 'State table' (Active: checked, Device: 4, Address: 10) and 'Command table' (Active: checked, Device: 4, Address: 20). At the bottom, there is a table titled 'CANopen devices:' with columns for Number, Node-ID, RxIndex, RxSub, TxIndex, and TxSub. A callout points to the first row of this table.

Number	Node-ID	RxIndex	RxSub	TxIndex	TxSub
1	5	29184	1	33280	1
2	8	29184	1	33280	1
3	6	29312	1	33408	1
4	15	29056	1	33152	1
5	24	29312	1	0	0
6	34	29056	1	0	0

↳ The protocol configuration defines the operator terminal in the CANopen environment. This configuration needs to know the network configuration, to deduce the NodeID and the speed. The NodeID characterises a device on the network and permits to configure the first SDO server. In the same way, we configure the timeout and the number of retries. We defines then if the operator sends an initialisation message to all the peripherals (send NMT start) or if it must wait for one (wait for NMT start). The 2 functions must not be validated in the same time. The option “Default SDO” can defines if we want to discuss with the first SDO server directly. In this case, you just have to specify the NodeID of each device of the CAN network which we want to communicate with. In the other case, you have to specify the reception COBID and the emission one of the SDO for the concerned device.

↳ A peripheral represents a place (for reading and writing) in the CANopen device dictionary. A peripheral characterises the access to a field which may correspond to a variable table. So 2 peripherals can belong to the same device (same NodeID or same COBID). A peripheral is characterised with the device it belongs to (NodeID or COBID), the index number and the basis number of the dictionary sub-index. Then you can give to a variable a peripheral number which corresponds to the peripheral table number and an address which corresponds to a sub-index shift. It means that the peripheral number fixes the linked device and the dictionary index. The sub-index is fixed by the addition of the sub-index defined in the peripheral table and the address of the variable. For example, we want to read 2 variables of a peripheral with a NodeID equal to 22. The locations to read are located to the index 7180h and to the sub-indexes 40h et 25h in the device dictionary. Some peripherals have already been declared, so our peripheral is number 6. In this case, we specify 34 in hexadecimal (22h) for the field NodeID of the peripheral number 6. We define for RxIndex the value 29056 in decimal (7180h) and for RxSub the value 1. The fields TxIndex and TxSub are dedicated to the writing, they have no use in our case. However we can specify them to values for writing variables.

Number	Name	Type	Size	Device	Adress	Initialize	Read	Write
1	VAR1	Numeric	Word	6	63	No	On show	None
2	VAR2	Numeric	Word	6	36	No	On show	None

For the 2 variables, we specify the number of their peripheral, which is 6. Then for each one we define its address, which is the shift regarding the sub-index. So for one we will have 63 (40h-1h= 3Fh) in decimal and the other 36 (25h-1h= 24h) in decimal. All these steps must be done to define the variables. You must have as many peripheral as there are indexes. We can have up to 16 peripherals.

↳ The state table and command table configuration needs the configuration of at least one peripheral in the peripheral table. Actually those two tables must be referenced regarding to a peripheral number. In the same way the address to specify is used as a shift value for the sub-index. The definition of those two tables is the same as in the ModBus protocol.

13-3-6- Example : CANopen link between an operator terminal and a MCS

The communication configuration between a MCS and a terminal consists in giving a NodeID to each one. A SDO communication from the terminal to the MCS is then possible. The default COBIDs of the SDO servers are 600h+NodeID in reception and 580h+NodeID in emission. We configure the respective clients according to that:

↳ Initialisation of the MCS

```
'Start of the board at 500KBits/s on the node 1
StartCan(Can1,1,5)
'COBID ClientSDO Rx Mcs = COBID ServerSDO Tx Terminal
```

CanSetup&(Can1,1280h,1,582h)

'COBID ClientSDO Tx Mcs = COBID ServerSDO Rx Terminal

CanSetup&(Can1,1280h,2,602h)

↳ Initialisation of the terminal

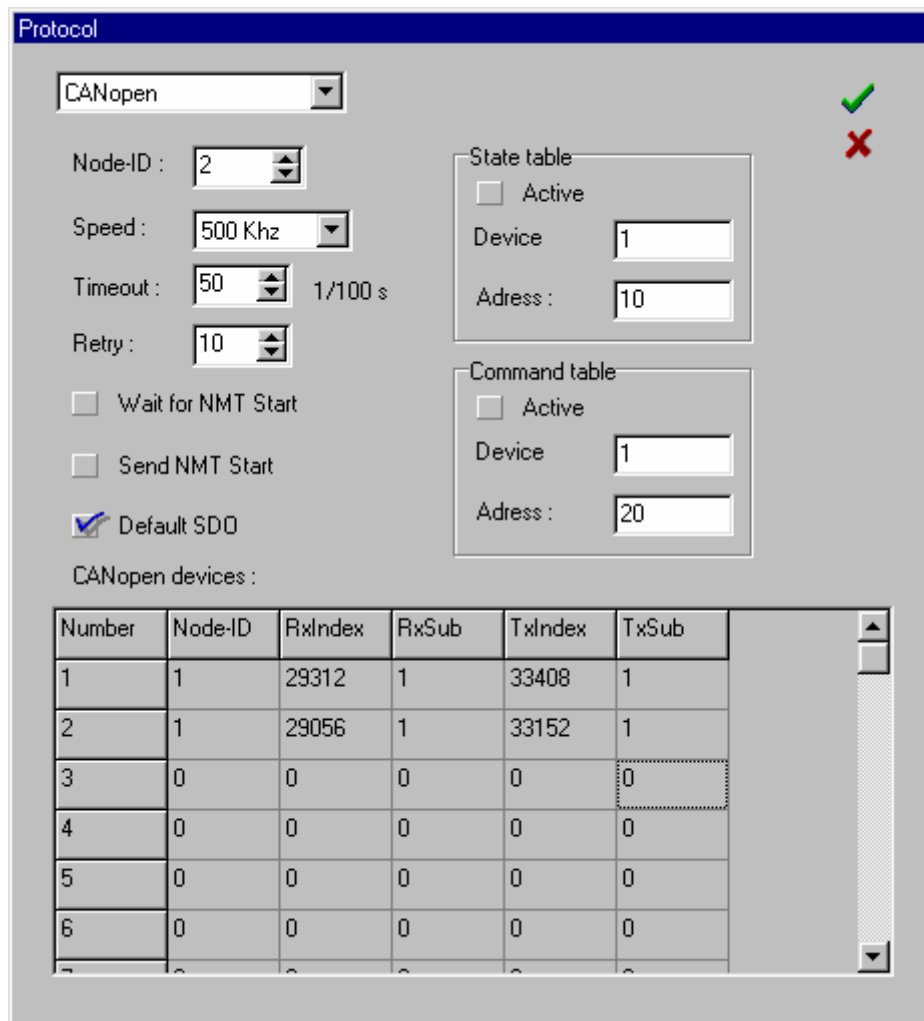
The peripherals 1 and 2 of the terminal are declared to access to the table “read and write of 16 bits non signed variables” and “read and write of 32 bits non signed variables”

The 2 peripherals are showing the same NodeID but the indexes and sub-indexes are different. When you choose to use the default SDO the terminal calculates the COBID regarding the NodeID. Then we associate every variable to a different CANopen table by associating variables to peripherals.

In this example, we used two variables in the communication between the operator terminal and the MCS32 EX. The first variable (VAR1) has the orders which are transmitted to the MCS. The second variable (VAR2) indicates the X axis position and will be displayed by the operator terminal.

The variable VAR1 must be written in the table “Write of 16 bits unsigned variables” of the MCS dictionary and the variable VAR2 must be read in the table “Read of 32 bits signed variables” of the MCS dictionary.

The configuration of the operator terminal is the next:



Number	Name	Type	Size	Device	Adress	Initialize	Read	Write
1	VAR1	Numeric	Word	1	1	No	Continue	On edit
2	VAR2	Numeric	DWord	2	1	No	Continue	On edit

The VAR2 variable will be displayed in a continue mode on the operator terminal screen. This variable will be modified by two different keys. A key press on the first key changes the state of the first bit of the VAR2 and a key press on the second key changes the state of the second bit of the VAR2.

The MCS uses these information like this:

```
O%=CanLocal%(Can1,1)           'Read the command
P&=RealToLong(Pos_S(X))
CanLocal&(Can1,1,P&)          'Read the position
If (O%=0) And (Move_S(X)=1) Then Stop(X)    'Stop if no command
If (O%=1) And (Move_S(X)=0) Then Stti(X=+)  'Bit 1 = Jog+
If (O%=2) And (Move_S(X)=0) Then Stti(X=-)  'Bit 2 = Jog-
```

Index

A

Adding and configuring an alarm.....	47
Alphanumerical variable display.....	26
Alphanumerical variable setting.....	28

C

Cancelling an alarm.....	48
Change the back screen colour.....	35
Characteristics.....	55
Configuration.....	56, 57

D

Declaration and modification of a variable.....	37
Declaration of a new page.....	21
Declaration of a program.....	44
Declaration of the ModBus.....	50
Definition of the protocol.....	50
Deletion of a page.....	23
Deletion of a program.....	45
Deletion of a variable.....	38
Dialog 640 presentation.....	5
Dialog 80 presentation.....	4
Dialog640 connection.....	10
Dialog640 mounting.....	11
Dialog80 connection.....	8
Dialog80 mounting.....	9
Dictionary.....	55, 56
Directories.....	13
DWIN software presentation.....	6
Dynamic text.....	24, 25, 26

E

Environmental consideration.....	8
Example CANopen link between an operator terminal and a MCS.....	57

F

File menu.....	14
Fonts.....	34

G

General presentation.....	36, 37, 39, 44, 46, 47, 49
---------------------------	----------------------------

H

Help menu.....	17
Horizontal line display.....	34

I

Initial screen.....	13
Installation procedure.....	12
Introduction.....	53

M

Main presentation.....	4
Mounting of the cell.....	9, 11

N

Numerical value display.....26
Numerical variable setting27

O

Options menu 17

P

Page21
Personalisable labels..... 10, 11
Presentation of the command table.....52
Presentation of the editor 23, 32
Presentation of the state table.....50
Procedure to receive programs 19
Procedure to send programs 19
Procedure to transfer a project..... 19
Programming of the dynamic keys39
Programming of the ESC key32, 41
Programming of the led function keys30
Programming of the simple dynamical keys29
Programming the LED function keys41
Project contents 13

R

Rectangle display34

S

Safety menu 16
Static text.....24
System Boot menu description..... 18
System configuration 12
System setup menu description 18

T

The CANopen communication54

U

Upgrade from previous version 18

V

Vertical line display34
View menu 16