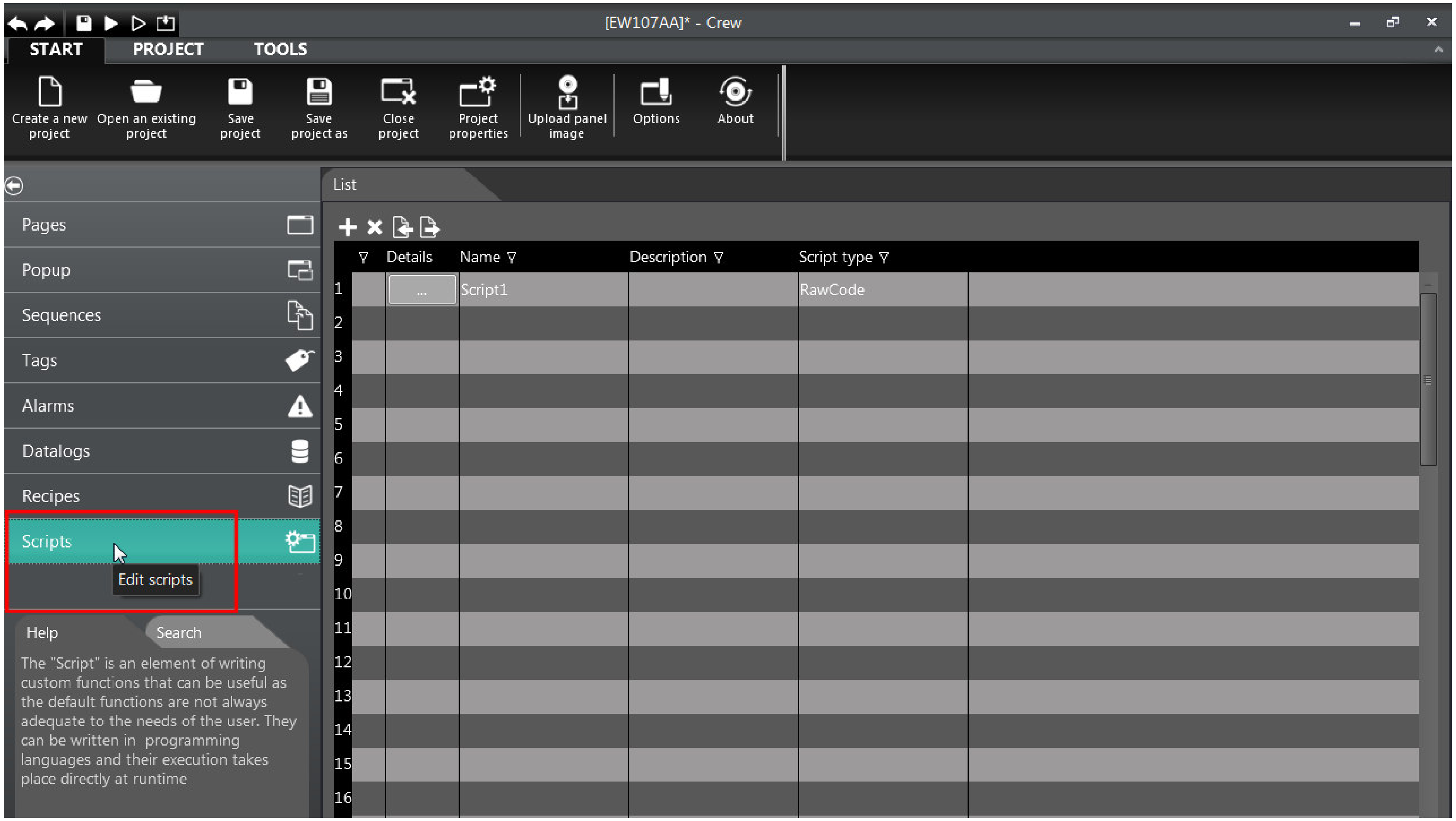
Scripts

CREW allows the programmer to add to the project whole programmes or functions for managing and editing all the application's components (graphic objects, variables, recipes etc.) in runtime.

Thanks to this, users can complement the set of predefined functions supplied by CREW with those they have created according to their needs.

User scripts can be called up in a project when a button is pressed, when an event is triggered or in response to being called by other scripts.

Scripts can be inserted into a project using Project Explorer :



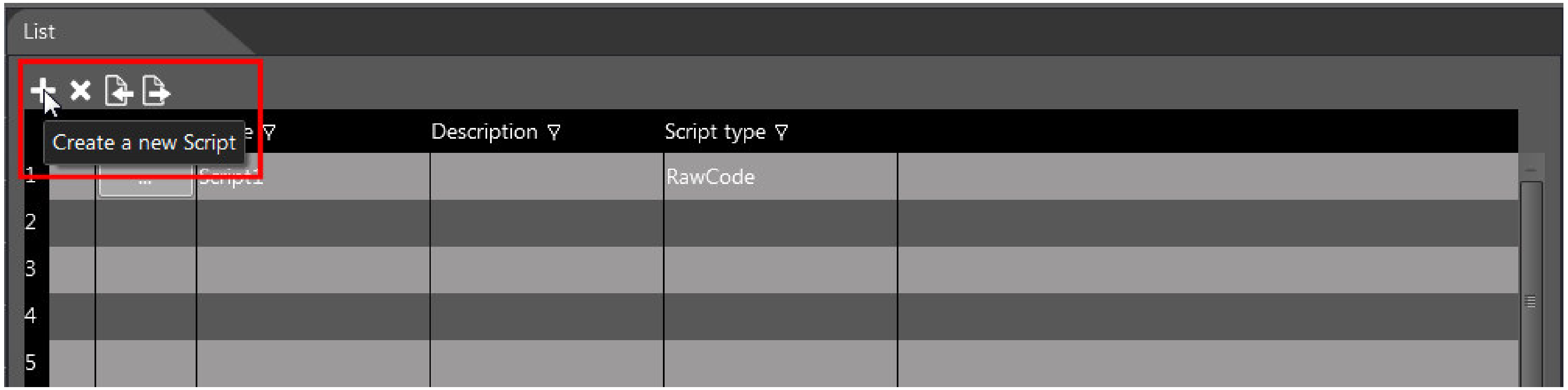
The "Script" is an element of writing custom functions that can be useful as the default functions are not always adequate to the needs of the user. They can be written in programming languages and their execution takes place directly at runtime

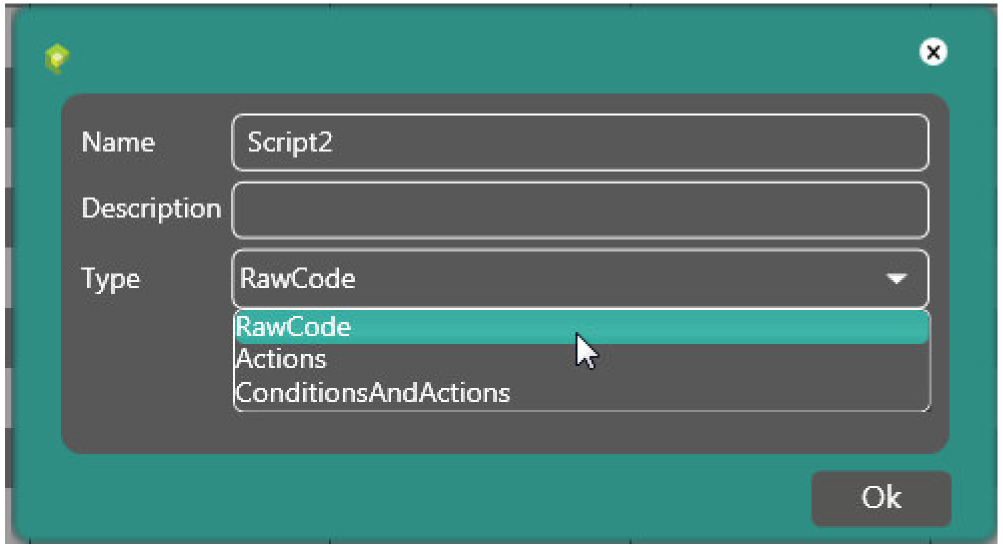
	Details	Name	Description	Script type
1	...	Script1		RawCode
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				

Their code can be written using simple programming/scripting languages like VBScript.

For details concerning programming techniques (variable declarations, operators, conditional structures and predefined functions) the user is advised to consult specialist manuals relating to the language to be used.

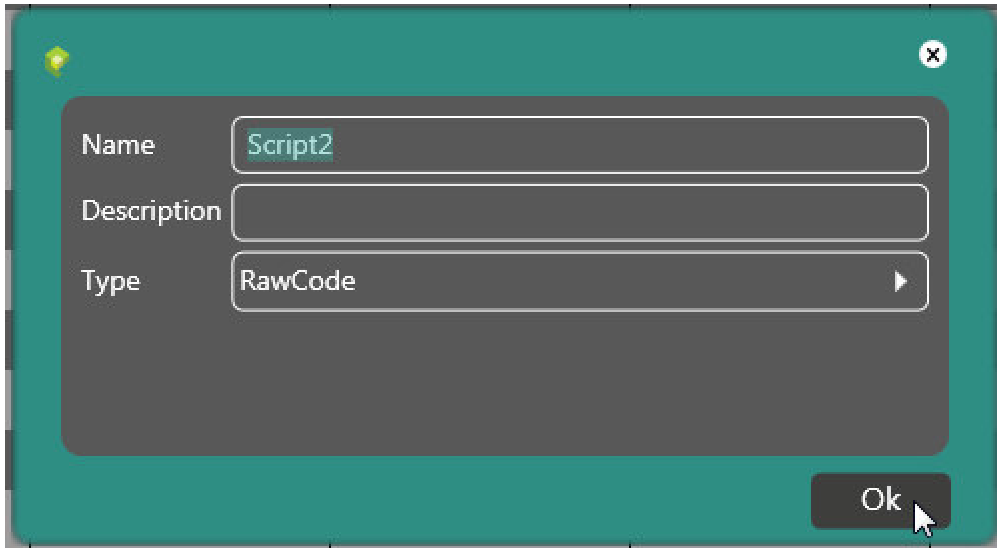
Creating a new script :




Name: Script2

Description:

Type: RawCode

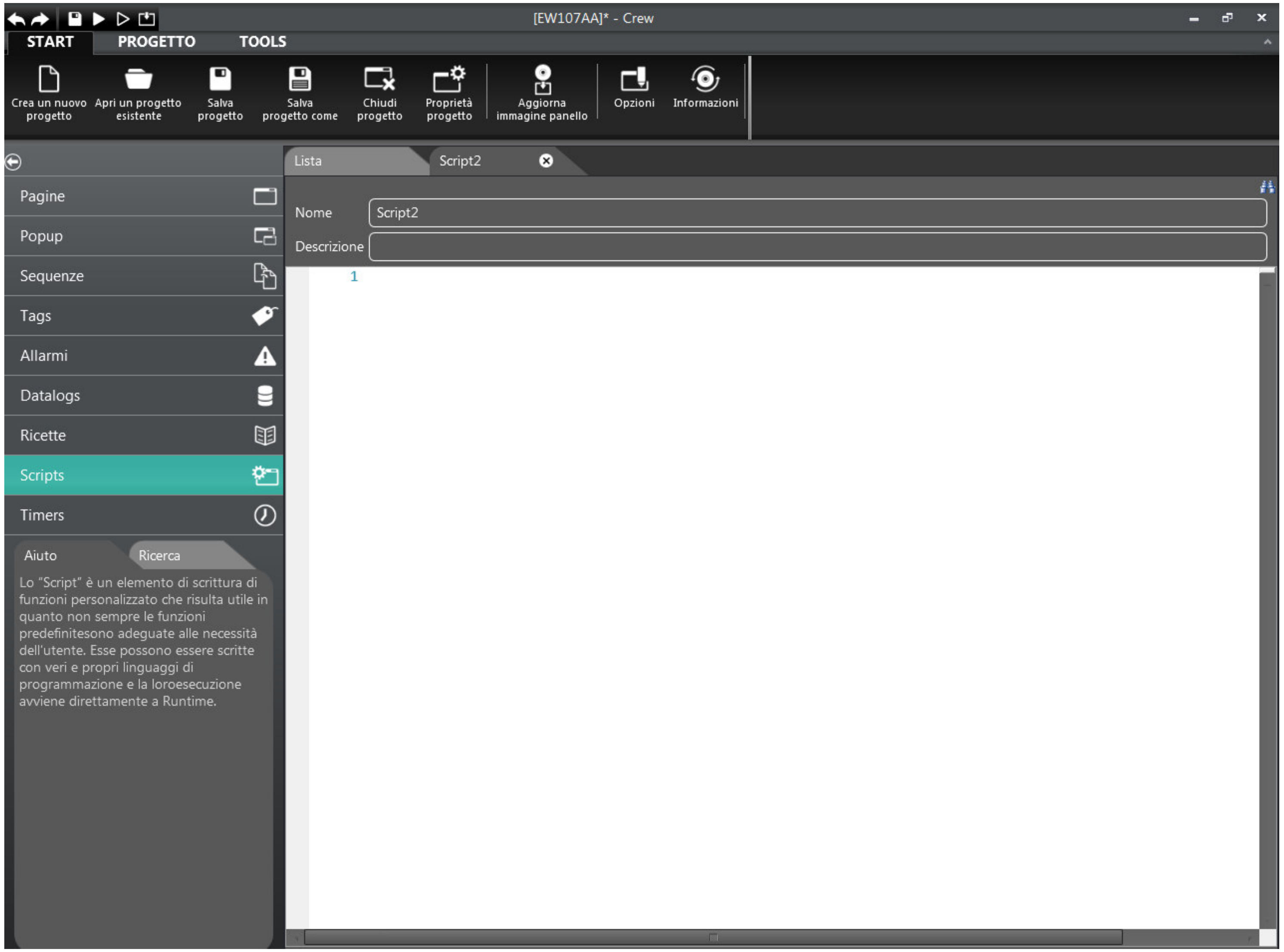

Name: Script2

Description:

Type: RawCode

In CREW once a script has been inserted using Project Explorer, the editor page for writing the code can be used.

The editor runs a real-time check of the syntax of the code, immediately posting an on-screen warning should it detect any imprecision in the formulation of the instructions :



Nome: Script2

Descrizione:

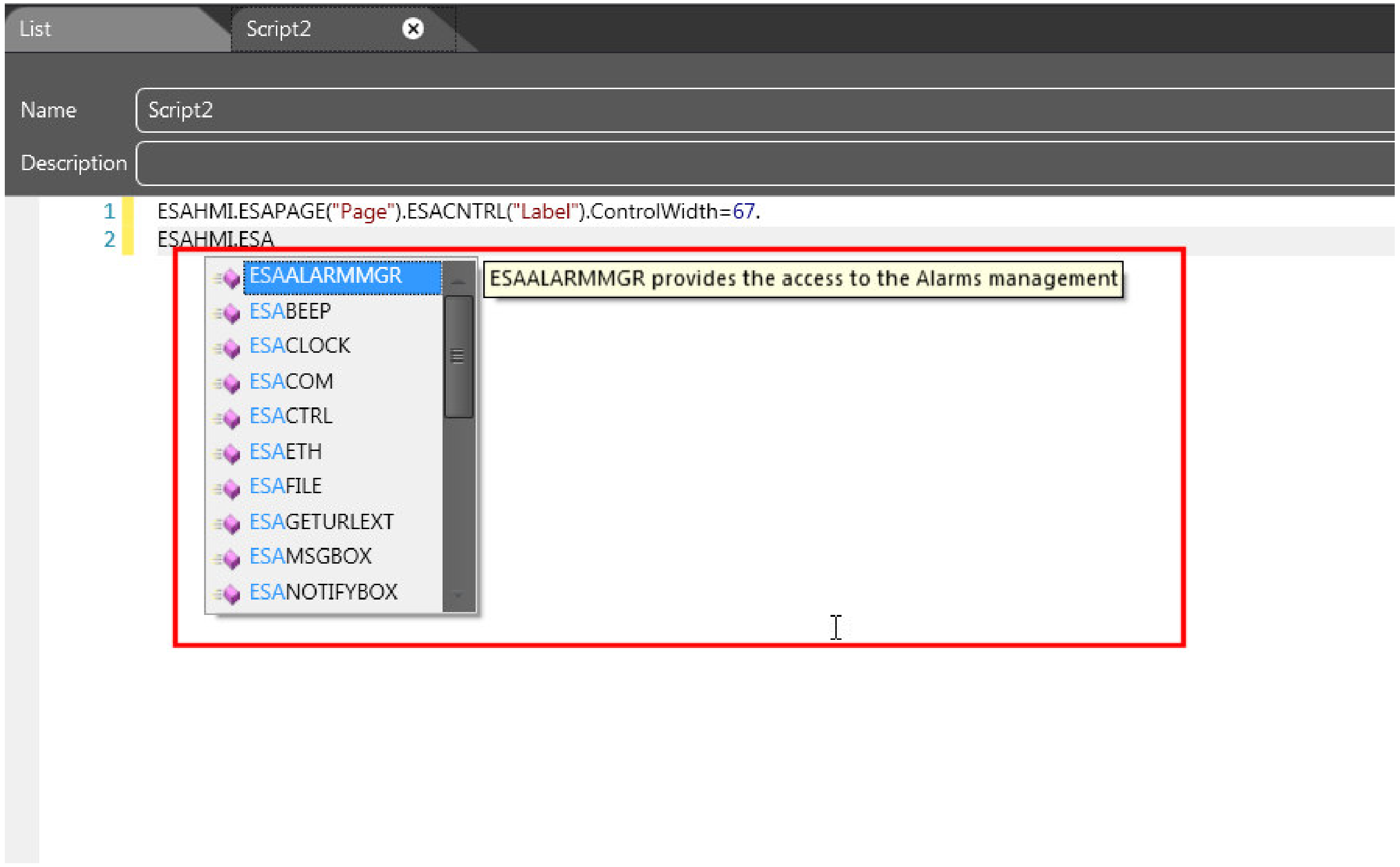
Lo "Script" è un elemento di scrittura di funzioni personalizzato che risulta utile in quanto non sempre le funzioni predefinitesono adeguate alle necessità dell'utente. Esse possono essere scritte con veri e propri linguaggi di programmazione e la loroesecuzione avviene direttamente a Runtime.

When the mouse cursor is placed on it, a complete description of the problem is put on screen.

The errors and their related descriptions are also listed the moment the project is validated and compiled.

The editor facilities the drafting of the code, showing too the list of objects and properties available for the object that has been inserted (Intellisense mechanism).

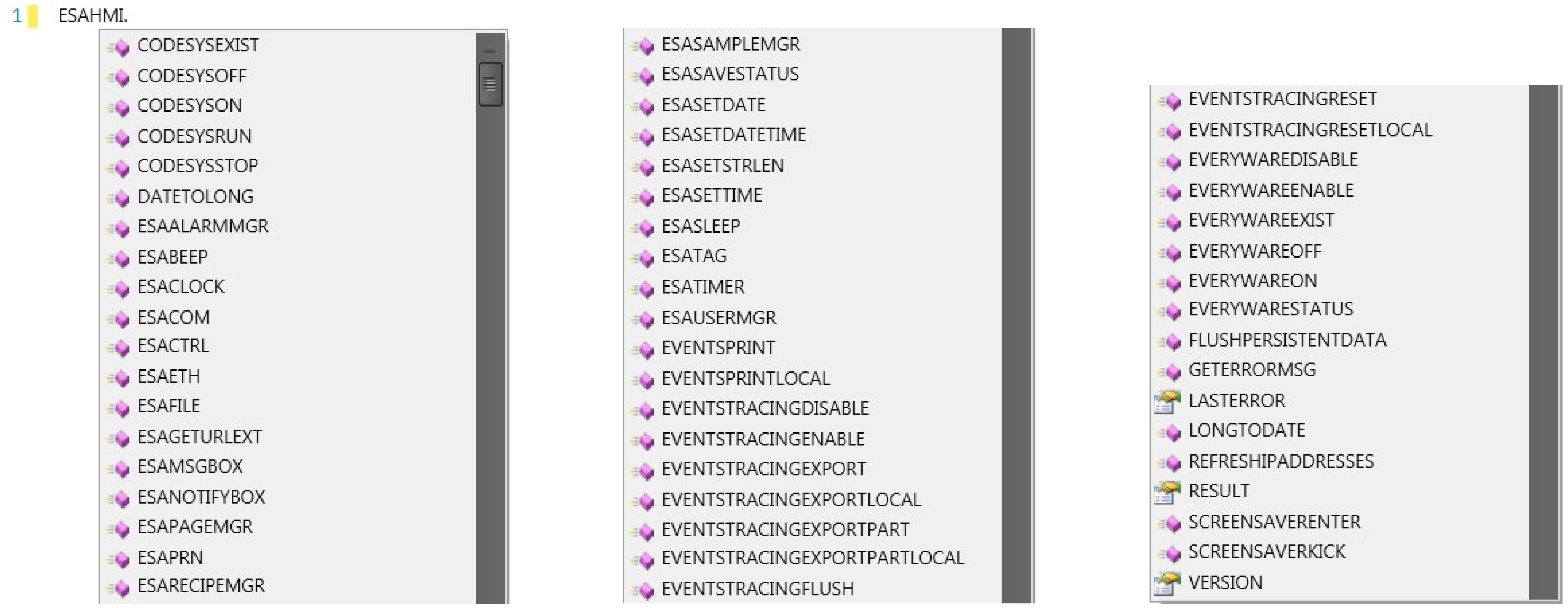
This list appears whenever the user presses the separation point between the objects or between an object and the method (or property) to be called :



```
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").ControlWidth=67.
ESAHMI.ESA
```

ESAALARMMGR provides the access to the Alarms management

When the code is edited, the objects are, in fact, separated from their respective children or methods by the insertion of a point '.' (dot). There follows a chart showing the hierarchy of objects accessible by script :



ESAHMI.

CODESYSEXIST
CODESYSOFF
CODESYSON
CODESYSRUN
CODESYSSTOP
DATETOLONG
ESAALARMMGR
ESABEEP
ESACLOCK
ESACOM
ESACTRL
ESAETH
ESAFILE
ESAGETURLEXT
ESAMSGBOX
ESANOTIFYBOX
ESAPAGEMGR
ESAPRN
ESARECIPEMGR
ESASAMPLEMGR
ESASAVESTATUS
ESASETDATE
ESASETDATETIME
ESASETSTRLEN
ESASETTIME
ESASLEEP
ESATAG
ESATIMER
ESAUSERMGR
EVENTSPRINT
EVENTSPRINTLOCAL
EVENTSTRACINGDISABLE
EVENTSTRACINGENABLE
EVENTSTRACINGEXPORT
EVENTSTRACINGEXPORTLOCAL
EVENTSTRACINGEXPORTPART
EVENTSTRACINGEXPORTPARTLOCAL
EVENTSTRACINGFLUSH
EVENTSTRACINGRESET
EVENTSTRACINGRESETLOCAL
EVERYWAREDISABLE
EVERYWAREENABLE
EVERYWAREEXIST
EVERYWAREOFF
EVERYWAREON
EVERYWARESTATUS
FLUSHPERSISTENTDATA
GETERRORMSG
LASTERROR
LONGTODATE
REFRESHIPADDRESSES
RESULT
SCREENSAVERENTER
SCREENSAVERKICK
VERSION

Therefore, to indicate an element of the page, we use an instruction of the type :

ESAHMI.ESAPAGE("Page").**ESACNTRL**("Label").ControlWidth=67.

In the case of those objects that require a passage of the name of the reference object (for example, ESAPAGE, ESACNTRL, etc.), after the opening of the brackets just press the '?' key of the keyboard to obtain the list of objects that can be inserted.

The following sections of this chapter will deal with the various objects accessible by Script and set out their properties and functions.

Note : Some properties mentioned in the following paragraphs are described as being in read-only mode when using Scripts; for many of these properties, however, there is no physical protection, so there is the possibility that the script will overwrite their value. This overwrite operation is, in any case, not advised. It is thus the programmer's responsibility to avoid the properties indicated as being read-only (R) being edited by the scripts.

Key to types of variable and syntactical premises

The following sections will refer to properties and methods characteristic of objects. The table below gives a rapid key to the abbreviations that will be used :

Variable	Abbreviation
Whole	Int
String	Str
Boolean	Bool
Long	Long
Double	Dbl
RGB (color, returned by the RGB function)	RGB
Variant	Var
R	Read, read-only
RW	Read&Write, read and write

If a subroutine (method returning no value) requires an input parameter, the passage can be achieved by using brackets or by leaving them out :

ESAHMI.ESAMSGBOX “Text”

ESAHMI.ESAMSGBOX(“Text”)

When a subroutine requires more than one input parameter, these parameters must be written consecutively and separated by a comma (without brackets) as shown in the following example :

ESAHMI.ESAPAGEMGR.ShowPageByNumber 32,0

If a function (a method returning a value) requires one of more input parameters, a passage must be made using brackets, as follows :

a=**ESAHMI.ESATAG**("Tag_Array").GetTagBitValue(1)

a=**ESAHMI.ESAPAGEMGR**.GetTAGBuffer (“RecipeType”, “RecipeName”)

Notes

Case

The names of all methods and properties are NOT case sensitive.

Example:

ESAHMI.ESAMSGBOX 123

Is equivalent to:

ESAhmi.ESAmsgbox 123

Subroutines

When not specified, the method returns no value (soubroutine)

Functions returning boolean values

FALSE has to be treated as “zero” (= 0).

TRUE has to be treated as “any value different from zero” (<> 0).

Parameters

When not specified , the methods parameters are input-parameters.
Output-parameters are expressly indicated.

Boolean parameters

FALSE has to be passed as “zero” (0).

TRUE has to be passed as “one” (1).

Optional Parameters

Optional parameters are indicated between brackets (ex: [suspensive]).

The default value of the suspensive parameter is defined in the Configurator at design-time.

“Local” Functions

There are couples of functions defined “Local” or “Global” (i.e. EventsPrintLocal and EventsPrint).

“Local” functions act at client-side while “Global” functions act at server-side

ESAHMI

ESAHMI is the main ESA object.

ESAHMI provides several utility methods and carries out the access to the ESA sub-objects.

Sub-Objects

ESAALARMMGR

ESACOM

ESACTRL

ESAETH

ESAFILE

ESAPAGEMGR

ESAPRN

ESARECIPEMGR

ESASAMPLEMGR

ESATAG

ESATIMER

ESAUSERMGR

Common Properties

The following properties are common to all the ESA objects :

Name	Type	Read-Write	Description
Version	integer	R	Object release version

Examples :

v = **ESAHMI.ESAFILE.version**

set obj = **ESAHMI.ESAFILE**

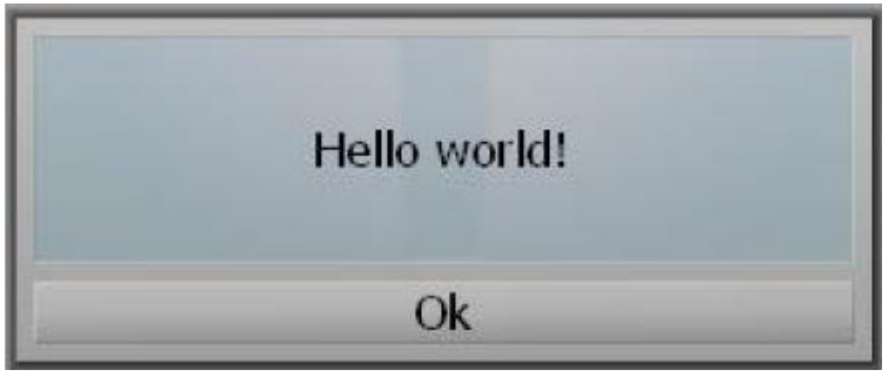
v = **obj.version**

Methods - ESAHMI

ESAMsgBox(message, [suspensive])

Message	(variant)	text of the message
suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays a message box



Example :

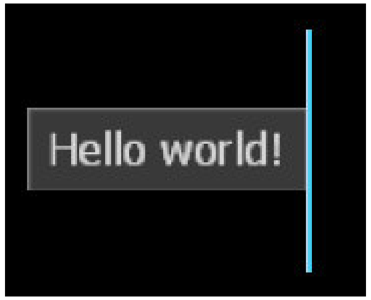
ESAHMI.ESAMSGBOX "Hello world!"

ESAHMI.ESAMSGBOX 1234.56

ESANotifyBox(message)

message	(variant)	text of the message

Displays a sliding notification box



ESASleep(msec)

msec	(integer)	milliseconds

Suspends the execution of the script for a specified interval.

ESABeep(frequency, duration)

frequency	(integer)	sound frequency (Hertz)
duration	(integer)	sound duration (milliseconds)

Generates simple tones on the speaker

ESASaveStatus()

Flushes the Windows registry.

ESASetDate(Day, Month, Year)

Day	(integer)	Day
Month	(integer)	Month
Year	(integer)	Year

Sets the system date (valid range: 1970-2105).

ESASetTime(Hour, Minutes, Seconds)

Hour	(integer)	Hour
Minutes	(integer)	Minutes
Seconds	(integer)	Seconds

Sets the system time

ESASetDateTime(date)

Date (DATE) Date & Time

Sets both system date and time

Example :

```
‘ set date value

v = datevalue("28/12/2013") or
v = dateserial(2013,12,28) or
v = CDATE("28/12/2013")
‘ set time value
v = timevalue("21:34:56") or
v = timeserial(21,34,56) or
v = CDATE("21:34:56")
‘ set date and time value
v = CDATE("04/10/2013 17:30:11")
‘ set system date and time
ESAHMI.ESASetDateTime v
‘ display date and time
d = day(v)
```

```
m = month(v)
y = year(v)
h = hour(v)
p = minute(v)
s = second(v)

ESAHMIESAmsgbox d &"-" &m &"-" &y &" " &h &":" &p &":" &s
```

FlushPersistentData()

Saves on disk all the persistent data.

ESASetStrLen(String, Character, Len, Right)

String	(string)	original string (can be empty)
Character	(string)	filling character (the first character is used)
Len	(integer)	final lenght of the string.

If len is greater than the length of the original string (String), the final string is filled using Character, otherwise the original string is truncated to the first Len characters

Right (boolean) : TRUE = the filling characters are appended at the end of the original string
FALSE = the filling characters are inserted at the beginning of the original string

Builds a string from another string. The new string is padded or truncated.

RETURN (string) : the new string

Example :

```
        s = ESAHMI.ESASetStrLen( "hello", "#", 10, 1)
```

ESAClock()

RETURN (integer) : the number of milliseconds since the device booted.

DateToLong(date)

Date (DATE) Date & Time

Converts a VBS-DATE into a LONG date

RETURN (integer) : the date in the LONG format
(number of seconds passed from 01-01-1970 00:00:00)

Example :

```
        t = CDATE("04/10/2013 17:30:11")
        v = ESAHMI.DateToLong(t)
```

LongToDate(Date)

Date (integer) Date & Time

Converts a LONG date into a VBS-DATE

RETURN (DATE) : the date in the VBS-DATE format

Example :

```
        t = ESAHMI.LongToDate(value)
        d = day(t)
        m = month(t)
        y = year(t)
```

GetErrorMsg(error)

error (integer) System error code

RETURN (string) : the message corresponding to the specified error code

Example :

```
        error = ESAHMI.LastError

        str = ESAHMI.GetErrorMsg(error)
        ESAHMI.ESAmsgbox str
```

ESAGetUrlExt(Url, Dest, Proxy, ProxyUsername, ProxyPassword, ServerPort, ServerUsername, ServerPassword)

Url (input parameter, string): address of file to be transferred

Dest (input parameter, string): full path of destination file

Proxy (input parameter, string): IP address of proxy:port (can be empty string)

ProxyUsername (input parameter, string): user name used for proxy authentication process (can be empty if proxy is not used)

ProxyPassword (input parameter, string): password used for proxy authentication process (can be empty if proxy

is not used)
ServerPort (input parameter, integer) : server port (default=80)
ServerUsername (input parameter, string): user name used for server authentication process (can be empty if server is not used)
ServerPassword (input parameter, string): password used for server authentication process (can be empty if server is not used)
Gets a file from an Url.

Example:

```
ESAHMI.ESAGetUrlExt "http://198.168.100.1/image.jpg", "\picture1.jpg",
                    "proxy:8080", "PROXYUSER", "PROXYPASSWORD",
                    5001, "admin", "admin"
```

ScreenSaverEnter()
Activates the screen-saver (the screen-saver must be enabled in the project).

ScreenSaverKick()
Resets the screen-saver timeout.

RefreshIpAddresses()
Updates the system variables with the list of the active IP addresses.

EventsTracingEnable()
Enables the FDA tracking.

EventsTracingDisable ()

Disables the FDA tracking.

EventsTracingFlush()

Saves on disk all the FDA tracking data.

EventsTracingExport(Pathname)

Pathname (string) Full path file name.

Exports on text file all the FDA tracking data.

RETURN (integer) : the number of exported records.

EventsTracingExportLocal(Pathname, [suspensive])

Pathname (string) Full path file name.
If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive (boolean-optional) TRUE = suspensive dialog box
FALSE = non-suspensive dialog box

Exports on text file all the FDA tracking data.

RETURN (integer) : the number of exported records.

EventsTracingExportPart(Pathname, From, To)

Pathname (string) Full path file name.

From (DATE) Starting date/time

To (DATE) Ending date/time

Exports on text file the FDA tracking data between dates/times.

RETURN (integer) : the number of exported records.

Example :

```
t1= CDATE("04/10/2013 08:00:00")
t2 = CDATE("04/10/2013 20:00:00")
n = ESAHMI.EventsTracingExportPart( "\data.txt", t1, t2 )
```

EventsTracingExportPartLocal(Pathname, From, To, [suspensive])

Pathname (string) Full path file name.

If Pathname is an empty string (""), a dialog box that allows the user to select a path is displayed.

From (DATE) Starting date/time

To (DATE) Ending date/time

suspensive (boolean-optional) TRUE = suspensive dialog box

FALSE = non-suspensive dialog box

Exports on text file the FDA tracking data between dates/times.

RETURN (integer) : the number of exported records.

Example :

```
t1= CDATE("04/10/2013 08:00:00")
```

```
t2 = CDATE("04/10/2013 20:00:00")
n = ESAHMI.EventsTracingExportPartLocal( "", t1, t2 )
```

EventsTracingReset(Pathname)

Pathname (string) Full path file name
Exports on text file all the FDA tracking data and clears the internal buffer.

EventsTracingResetLocal(Pathname, [suspensive])

Pathname (string) Full path file name

If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive (boolean-optional) TRUE = suspensive dialog box
FALSE = non-suspensive dialog box

Exports on text file all the FDA tracking data and clears the internal buffer.

EventsPrint()

EventsPrintLocal()

Prints the FDA tracking data.

RETURN (integer) : The number of printed events.

EverywareOn()

EverywareOff()

Activate/Inactivate EveryWare process

EverywareEnable()

EverywareDisable()

Enable/Disable EveryWare process.

EverywareStatus()

Gets the EveryWare status.
RETURN (integer) : 0 = process not active (!0 = process active)
1= process enabled
2 = process disabled
... = process enabled, connection error

EverywareExist()

Gets the EveryWare status.

RETURN (integer) : TRUE = process active
FALSE = process not active

CoDeSysOn()

CoDeSysOff()

Activate/Inactivate CoDeSys process.

CoDeSysRun()
CoDeSysStop()

Start/Stop CoDeSys application.

CoDeSysExist()

Gets the CoDeSys status.

RETURN (integer) : TRUE = process active
FALSE = process not active

Properties

Name	Type	Read-Write	Description
Result	integer	R	Result of the last ESA call (0 = no error)
LastError	integer	R	System error of the last ESA call (0 = no error)

Examples:

```
Sub test1
        on error resume next
        ESAHMI.ESAFILE.Open "filename","r"
        res = ESAHMI.Result
        if res <> 0 then
                syserr = ESAHMI.LastError
                strerr = ESAHMI.GetErrorMsg(syserr)
                ESAHMI.ESAmsgbox "result=" & HEX(res) & " - " & syserr & " : " & strerr
        end if
        ...
Sub test2
        on error resume next
        ESAHMI.ESAFILE.Open "filename","r"
```

```
if err.number <> 0 then
        syserr = ESAHMI.LastError
        strerr = ESAHMI.GetErrorMsg(syserr)
        ESAHMI.ESAmsgbox "result=" & HEX(err.number) & " - " & syserr & " : " & strerr
end if
...
```

ESAALARMMGR

ESAALARMMGR provides the access to the Alarms management.

Methods - ESAALARMMGR

If UserName is an empty string “”, the current user name is used.
If StationName is an empty string “”, the local station name is used.

AlarmOn(AlarmName, UserName, StationName)

AlarmName	(string)	Name of the alarm to raise
UserName	(string)	Name of the user who raises the alarm
StationName	(string)	Name of the station where the alarm is raised

Raises the specified alarm.

RETURN (integer) : the alarm Instance identified (used by AckAlarm)

ClearAlarm(AlarmName, UserName, StationName)

AlarmName	(string)	Name of the alarm to clear
UserName	(string)	Name of the user who raises the alarm
StationName	(string)	Name of the station where the alarm is raised

Clears an alarm.

AckAlarm(Instance, UserName, StationName)

Instance	(integer)	Instance of the needed alarm (returned from AlarmOn)
UserName	(string)	Name of the user requesting the action
StationName	(string)	Name of the station requesting the action

Acknowledges a single instance of an alarm.

AckInstances(AlarmName, UserName, StationName)

AlarmName	(string)	Name of the alarm to acknowledge
UserName	(string)	Name of the user requesting the action
StationName	(string)	Name of the station requesting the action

Acknowledges all the instances of the specified alarm.

AckGroup(GroupName, UserName, StationName)

GroupName	(string)	Name of alarms group
UserName	(string)	Name of user requesting the action
StationName	(string)	Name of the station requesting the action

Acknowledges all the alarms of specified group.

AckGlobal(UserName, StationName)

UserName	(string)	Name of user requesting the action
StationName	(string)	Name of the station requesting the action

Acknowledges all the active alarms.

IsAlarmOn(AlarmName)

AlarmName	(string)	Name of the alarm to check
RETURN	(boolean) : TRUE if the specified alarm is ON	

AlarmsExport(PathName)

Pathname	(string)	Full path file name.

Exports in a file the description of all the currently active alarms.

RETURN (integer) : the number of exported records.

AlarmsExportLocal(PathName, [suspensive])

Pathname (string)	Full path file name. If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive (boolean-optional)	TRUE = suspensive dialog box
	FALSE = non-suspensive dialog box

Exports in a file the description of all the currently active alarms.
RETURN (integer) : the number of exported records.

HistoryExport(PathName)

Pathname (string) Full path file name.

Exports in a file the description of all the alarms logged in the history.

RETURN (integer) : the number of exported records.

HistoryExportLocal(PathName, [suspensive])

Pathname (string)	Full path file name. If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive (boolean-optional) TRUE = suspensive dialog box
FALSE = non-suspensive dialog box

Exports in a file the description of all the alarms logged in the history.
RETURN (integer) : the number of exported records.

HistoryDelete()
Clears the alarms history.

HistoryFlush()
Save on disk the alarm history data.

AlarmsPrint()
AlarmsPrintLocal()

Prints the currently active alarms.

RETURN (integer) : The number of printed alarms.

HistoryPrint()

HistoryPrintLocal()

Prints the alarms logged in the history.

RETURN (integer) : The number of printed alarms.

ESACOM([index])

index (integer-optional) port index (1..4 – default=1)

ESACOM provides the access to the serial COM ports.

Up to 4 COM ports are available at the same time.

There is no relation between the port index and the COM port number.

The following calls are equivalent:

ESAHMI.ESACOM.Open 2,9600,8,0,0

ESAHMI.ESACOM().Open 2,9600,8,0,0

ESAHMI.ESACOM(1).Open 2,9600,8,0,0

Methods - ESACOM([index])

Open(Port, Baud, DataBits, Parity, StopBits, [Rts], [Cts])

Port (integer) port number (1=COM1, ...)

Baud (integer) baud rate

DataBits (integer) 4 .. 8

Parity (integer) 0=NONE, 1=ODD, 2=EVEN, 3=MARK, 4=SPACE

StopBits (integer) 0=1, 1=1.5, 2=2

Rts (integer-optional) 0=DISABLE (default), 1=ENABLE, 2=HANDSHAKE, 3=TOGGLE

Cts (integer-optional) 0 .. 1(default=0)

Opens a serial port for reading/writing.

Up to 4 ports can be opened at a time.

Close()

Closes an opened serial port.

IsOpen()

Checks if the port is open.

RETURN (boolean) : TRUE if the port is open, otherwise FALSE

IsData()

Checks if there are data available.

RETURN (integer) : 0 = no bytes in queue
>0 = number of bytes in queue

WriteByte(byte)

byte (integer) byte value to write

Writes a single byte on the serial port.

WriteStr(text)

text (string) Text to write

Writes a text string on the serial port.

ReadByte()

Reads a single byte from the serial port.

RETURN (integer) : the byte read.

Clear()

Clears/Resets the serial port.

Escape(code)

code (integer)	1	Simulate XOFF received
	2	Simulate XON received
	3	Set RTS high
	4	Set RTS low
	5	Set DTR high
	6	Set DTR low
	7	Reset device if possible
	8	Set the device break line
	9	Clear the device break line

Performs an extended function on the serial port.

SetRTS()

Sets the RTS (request-to-send) signal.

ClrRTS()

Clears the RTS (request-to-send) signal.

GetCTS()

Checks the CTS (clear-to-send) signal.

Return (boolean) : TRUE = signal HI, FALSE = signal LOW

GetDSR()

Checks the DSR (data-set-ready) signal.

Return (boolean) : TRUE = signal HI, FALSE = signal LOW

GetRing()

Checks the Ring Indicator signal.

Return (boolean) : TRUE = signal HI, FALSE = signal LOW

GetRLSD()

Checks the RLSD (receive-line-signal-detect) signal.

Return (boolean) : TRUE = signal HI, FALSE = signal LOW

Examples

```
set ser = ESAHMI.ESACOM
ser.Open 1, 19200, 8, 0, 0
ser.WriteByte 97 ' a
ser.WriteByte &h61 ' a
ser.WriteByte ASC("a") ' a
ser.Close

set ser1 = ESAHMI.ESACOM(1)
set ser2 = ESAHMI.ESACOM(2)
ser1.Open 3, 9600, 8, 0, 0
ser2.Open 4, 9600, 8, 0, 0
ser1.WriteStr "com1"
ser2.WriteStr "com2"
ser1.Close
ser2.Close

s = ""
n = ser.IsData
for i=1 to n
a = ser.ReadByte
s = s & chr(a)
next
```

ESACTRL

ESACTRL provides the access to the visual objects displayed in an opened page.

ESACTRL (page, controlname)

page	(string/ integer) page name/id
controlname	(string) control name

Example 1(slower):

```
v1= ESAHMI.ESACTRL("page1","txt1").Left
v2 = ESAHMI.ESACTRL("page1","txt1").Top
ESAHMI.ESACTRL("page1","txt1").Left = v1+ 20
ESAHMI.ESACTRL("page1","txt1").Top = v2 + 10
```

Example 2 (faster, recommended):

```
set ctrl = ESAHMI.ESACTRL("page1","txt1")
v1= ctrl.Left
v2 = ctrl.Top
ctrl.Left = v1+ 20
ctrl.Top = v2 + 10
```

Methods - ESACTRL

SetRangeColor (index, normalcolor, gradientcolor)

index	(integer)	range index (1.32)
normalcolor	(integer)	RGB color
gradientcolor	(integer)	RGB color

Sets a range color of a BAR Control.

GetRangeColor (index, normalflag)

index	(integer)	range index (1.32)
normalflag	(boolean)	TRUE=normalcolor, FALSE=gradientcolor

Gets a range color of a BAR Control.

RETURN (integer) : RGB range color (BAR Control).

SetRangeValue (index, value)

index	(integer)	range index (1.32)
value	(double)	floating-point value

Sets a range value of a BAR Control.

GetRangeValue (index)

index	(integer)	range index (1..32)

Gets a range value of a BAR Control.

RETURN (double) : range value (BAR Control).

SetMoveState (index, left, top, angle, time)

index	(integer)	step index (1..16)
left	(double)	X coordinate
top	(double)	Y coordinate
angle	(double)	rotation angle (degrees)
time	(double)	step time (msec)

Sets a movement step.

GetMoveStateLeft (index)

GetMoveStateTop (index)

GetMoveStateAngle (index)

GetMoveStateTime (index)

index	(integer)	step index (1..16)

Gets a movement step data.

RETURN (double/integer) : movement step data

SetImage (index, imagename)

index	(integer)	image index (1...)
imagename	(string)	full-path image filename

Sets an image to an Image Control.

GetImage (index)

index	(integer)	image index (1...)

Gets an image from an Image Control.

RETURN (string) : full-path image filename

SetText (index, text)

index	(integer)	text index (1...)
text	(string)	text

Sets a text to a Text Control.

GetText (index)

index	(integer)	text index (1...)

Gets a text from a Text Control.

RETURN (string) : text string

SetPointCoord (index, normalcolor, gradientcolor)

index	(integer)	range index (1..16)
x	(double)	X coordinate
y	(double)	Y coordinate

Sets a coordinate point of an Interlines Control.

SetTrendTraceMinX (index, value)

SetTrendTraceMaxX (index, value)
SetTrendTraceMinY (index, value)
SetTrendTraceMaxY (index, value)

index	(integer)	trace index (1..32)
value	(double)	floating-point value

Sets a range value of a Trend Control.

GetTrendMinX (index)
GetTrendMaxX (index)
GetTrendMinY (index)
GetTrendMaxY (index)

index	(integer)	trace index (1..32)

Gets a range value of a Trend Control.

RETURN (double) : range value (BAR Control).

TraceImportLocal(Pathname, [suspensive])

Pathname	(string)	Full path file name. If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive	(boolean-optional)	TRUE = suspensive dialog box
		FALSE = non-suspensive dialog box

Imports tracking data from text file.

Methods Table – ESACTRL

	SHAPE	COMPLEX	LINE	RECTANGLE	ELLIPSE	PATH	IMAGE	TEXT	PIPE	BUTTON	INTERLINE	BAR	SLIDER	METER	TREND	GRID
SetImage							*									
GetImage							*									
SetText								*								
GetText								*								
SetPointCoord											*					
SetRangeColor												*				
GetRangeColor												*				
SetRangeValue												*				
GetRangeValue												*				
SetMoveState	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
GetMoveStateLeft	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
GetMoveStateTop	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
GetMoveStateAngle	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
GetMoveStateTime	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Properties Table - ESACTRL

Name	Type	Read-Write
AlphaLevel	byte	RW
AnimationTime	integer	RW
BarContinuousColors	boolean	RW
Brightness	double	RW
ClipMode	integer	RW
ControlType	integer	R
CornerDownLeft	double	RW
CornerDownRight	double	RW
CornerUpLeft	double	RW
CornerUpRight	double	RW
Disabled	boolean	RW
FillAreaBlinking	boolean	R
FillColorBlink	integer	RW
FillColorBlink Grad	integer	RW
FillColorGrad	integer	RW
FillColorNormal	integer	RW
Flashing	boolean	R
GridBorderColor	integer	RW
GridBorderColorGrad	integer	RW
GridCellAlternateColor	integer	RW
GridCellHeaderColor	integer	RW
GridCellNormalColor	integer	RW
GridCellSelectColor	integer	RW
GridEmptyAlpha	byte	RW
GridEmptyColor	integer	RW
GridTextHeaderColor	integer	RW
GridTextNormalColor	integer	RW
Height	double	RW
HorizontalAlignment	integer	RW
Hue	Double	RW
Left	double	RW
LineBlinking	boolean	R
LineColorBlink	integer	RW
LineColorBlinkGrad	integer	RW
LineColorGrad	integer	RW
LineColorNormal	integer	RW
LineSize	integer	RW
MoveOn	boolean	RW
RotationAngle	double	RW
RotationCenterX	double	RW
RotationCenterY	double	RW
Saturation	double	RW
ScaleValueMaximum	double	RW
ScaleValueMinimum	double	RW
ShowLayerDisabled	Boolean	RW
ShowLayerInvalid	Boolean	RW
ShowLayerOffline	Boolean	RW
ShowLayerProtected	Boolean	RW
StartAngle	double	RW
StretchMode	integer	RW
SweepAngle	double	RW
TextContentLength	integer	R
Top	double	RW
TrendAreaInColor	integer	RW
TrendAreaInColorGrad	integer	RW
TrendAreaOutColor	integer	RW
TrendAreaOutColorGrad	integer	RW
TrendBorderColor	integer	RW
TrendBorderColorGrad	integer	RW
TrendGridMode	boolean	RW
TrendHasArea	boolean	RW
TrendHasLines	boolean	RW
TrendHasMarkers	boolean	RW
TrendIsDigitalLine	boolean	RW
TrendIsHistogram	boolean	RW
TrendLabelsColor	integer	RW
TrendTimeMode	integer	RW
TrendTimeSpan	integer	RW
TrendTouchLeft	double	RW
TrendTouchRight	double	RW
TrendTouchTop	double	RW
TrendTouchBottom	double	RW
Value	variant	RW
ValueType	integer	R
VerticalAlignment	integer	RW
Visible	boolean	RW
VoidBlinkColor	integer	RW
VoidBlinkColorGrad	integer	RW
VoidBlinking	boolean	R
VoidColor	integer	RW
VoidColorGrad	integer	RW
Width	double	RW
X1	double	RW
X2	double	RW
Y1	double	RW
Y2	double	RW

	SHAPE	COMPLEX	LINE	RECTANGLE	ELLIPSE	PATH	IMAGE	TEXT	PIPE	BUTTON	INTERLINE	BAR	SLIDER	METER	TREND	GRID
ControlType	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Visible	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Flashing	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Disabled	*						*	*		*	*	*	*	*		
ValueType	*						*	*	*	*		*	*	*		
Value	*						*	*	*	*		*	*	*		
Left	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Top	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Width	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Height	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RotationAngle	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RotationCenterX	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RotationCenterY	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
MoveOn	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
AlphaLevel			*	*	*	*	*	*	*		*	*	*	*	*	*
FillColorNormal				*	*	*		*	*							
FillColorGrad				*	*	*		*	*							
FillAreaBlinking				*	*	*			*			*				
FillColorBlink				*	*	*			*			*				
FillColorBlinkGrad				*	*	*			*			*				
LineSize			*	*	*	*			*		*					
LineColorNormal			*	*	*	*		*	*		*					
LineColorGrad			*	*	*	*		*	*		*					
LineBlinking			*	*	*	*			*		*					
LineColorBlink			*	*	*	*			*		*					
LineColorBlinkGrad			*	*	*	*			*		*					
HorizontalAlignment							*	*								
VerticalAlignment							*	*								
AnimationTime	*						*	*								
X1			*													
X2			*													
Y1			*													
Y2			*													

	SHAPE	COMPLEX	LINE	RECTANGLE	ELLIPSE	PATH	IMAGE	TEXT	PIPE	BUTTON	INTERLINE	BAR	SLIDER	METER	TREND	GRID
CornerUpLeft				*												
CornerUpRight				*												
CornerDownLeft				*												
CornerDownRight				*												
SweepAngle					*											
StartAngle					*											
ClipMode					*											
StretchMode							*									
TextContentLength								*								
VoidColor									*			*				
VoidColorGrad									*			*				
VoidBlinking									*			*				
VoidBlinkColor									*			*				
VoidBlinkColorGrad									*			*				
BarContinuousColors												*				
TrendBorderColor															*	
TrendBorderColorGrad															*	
TrendAreaInColor															*	
TrendAreaInColorGrad															*	
TrendAreaOutColor															*	
TrendAreaOutColorGrad															*	
TrendLabelsColor															*	

	SHAPE	COMPLEX	LINE	RECTANGLE	ELLIPSE	PATH	IMAGE	TEXT	PIPE	BUTTON	INTERLINE	BAR	SLIDER	METER	TREND	GRID
TrendGridMode															*	
TrendTimeMode															*	
TrendTimeSpan															*	
TrendHasMarkers															*	
TrendHasLines															*	
TrendHasArea															*	
TrendIsDigitalLine															*	
TrendIsHistogram															*	
GridBorderColor																*
GridBorderColorGrad																*
GridEmptyColor																*
GridEmptyAlpha																*
GridTextNormalColor																*
GridTextHeaderColor																*
GridCellNormalColor																*
GridCellAlternateColor																*
GridCellSelectColor																*
GridCellHeaderColor																*
ScaleValueMinimum												*	*	*		
ScaleValueMaximum												*	*	*		
Brightness	*															
Hue	*															
Saturation	*															
ShowLayerDisabled	*						*	*		*	*	*	*	*		
ShowLayerOffline	*						*	*	*	*		*	*	*		
ShowLayerInvalid	*						*	*	*	*		*	*	*		
ShowLayerProtected	*						*	*	*	*	*	*	*	*	*	*

ESAETH

ESAETH provides the access to the Ethernet ports.

Sub-Objects :

TCPCLIENT
TCPSERVER
UDP

ESAETH.TCPCLIENT

TCPCLIENT provides the Client management in TCP protocol.

Methods :

Open(IPAddress, Port)

IPAddress	(string)	Server IP Address (“xxx.xxx.xxx.xxx”)
Port	(integer)	Server Port number

Open a TCP Client socket and connect it to a server specified by address and port.

RETURN (integer): The TCP Client socket identifier.

Close(SocketId)

SocketId	(integer)	TCP Client socket identifier

Close the TCP Client socket.

IsData(SocketId)

SocketId	(integer)	TCP Client socket identifier

Check if there are data available.

RETURN (boolean) : 0 = no bytes in queue
<>0 = number of bytes in queue

GetAddress(SocketId, IpAddress)

SocketId	(integer)	TCP Client socket identifier
IPAddress	(string - output)	Client IP Address

Get port and address of the specified Client.

RETURN (integer) : The Client Port.

GetServerAddress(SocketId, IpAddress, serverPort)

SocketId	(integer)	TCP Client socket identifier
IPAddress	(integer – output)	Server IP Address

Get port and address of the server to which the specified client is connected.

RETURN (integer) : The Server Port.

WriteString(SocketId, String)

SocketId	(integer)	TCP Client socket identifier
String	(string)	String to send

Send a characters string to the TCP server.

RETURN (integer) : The number of characters written.

WriteBuffer(SocketId, Buffer, NumBytes)

SocketId	(integer)	TCP Client socket identifier
Buffer	(array of bytes)	Binary data to send
NumBytes	(integer)	Number of bytes to send

Send binary data to the TCP server.

RETURN (integer) : The number of bytes written.

ReadString(SocketId, String, NumChars)

SocketId	(integer)	TCP Client socket identifier
String	(string – output)	Received string
NumChars	(integer)	Number of characters to read (0 = read all available)

Receive a characters string from a TCP Client socket.

RETURN (integer) : The number of characters read.

ReadBuffer(SocketId, Buffer, NumBytes)

SocketId	(integer)	TCP Client socket identifier
Buffer	(array of bytes – output)	Received data

NumBytes (integer) Number of bytes to read (0 = read all available)

Receive binary data from a TCP Client socket.

RETURN (integer) : The number of bytes read.

Example 1

Send and receive data string:

```
serverIPAddress = "192.168.1.20"
serverIPPort = 20000
TCPClientID = ESAHMI.ESAETH.TCPClient.OPEN( serverIPAddress, serverIPPort )

result = ESAHMI.ESAETH.TCPClient.ISDATA(TCPClientID)
if result <> 0 then
        result = ESAHMI.ESAETH.TCPClient.READSTRING(TCPClientID,stringReceived,10)
        ESAHMI.ESAmsgbox "Received " + CStr(result) + " character(s) from " +
                serverIPAddress + ":" + CStr(serverIPPort) + " " +
                stringReceived
end if
result = ESAHMI.ESAETH.TCPClient.WRITESTRING(TCPClientID,"example")
ESAHMI.ESAETH.TCPClient.CLOSE TCPClientID
```

Example 2

Send and receive array of bytes:

```
serverIPAddress = "192.168.1.20"
serverIPPort = 20000
TCPClientID = ESAHMI.ESAETH.TCPClient.OPEN( serverIPAddress, serverIPPort )

result = ESAHMI.ESAETH.TCPClient.ISDATA(TCPClientID)
if result <> 0 then
        Dim dataReceived
        bytesRead = ESAHMI.ESAETH.TCPClient.READBUFFER(TCPClientID,dataReceived,10)
        For i = LBound(dataReceived) To UBound(dataReceived)
        ESAHMI.ESAMsgBox "Bytes read (" & CStr(bytesRead) & ") from " & CStr(serverIPAddress) & ":" &
        CStr(serverIPPort) & VBCrLf & dataReceived(i)
        Next
end if

'create an array of 13 bytes
Dim dataToSend
ReDim dataToSend(12)
'initialize array of bytes
For kk = LBound(dataToSend) To UBound(dataToSend)
```

```
        dataToSend(kk) = CByte(kk)
Next

bytesWritten = ESAHMI.ESAETH.TCPClient.WRITEBUFFER(TCPClientID,dataToSend,10)
ESAHMI.ESAmsgbox "Written " + CStr(result) + " byte(s)"
ESAHMI.ESAETH.TCPClient.CLOSE TCPClientID
```

ESAETH.TCPSERVER

TCPSERVER provides the Server management in TCP protocol.

Methods :

Open(IPAddress, Port)

IPAddress (string) Server IP Address ("xxx.xxx.xxx.xxx")

Port (integer) Server Port number

Open a TCP Server socket.

RETURN (integer): The TCP Server socket identifier.

Close(SocketId)

SocketId (integer) TCP Server socket identifier

Close the TCP Server socket.

CloseClient(SocketId)

SocketId (integer) TCP Server socket identifier

IPAddress (string) Client IP Address ("xxx.xxx.xxx.xxx")

Port (integer) Client Port number

Close the connection between the specified Client and the Server.

IsData(SocketId, IPAddress, Port)

SocketId (integer) TCP Server socket identifier

IPAddress (string – output) Client IP Address (“xxx.xxx.xxx.xxx”)

Port (integer – output) Client Port number

Checks if there are data available.

RETURN (boolean) : 0 = no bytes in queue

<>0 = number of bytes in queue

IsDataClient(SocketId, IPAddress, Port)

SocketId (integer) TCP Server socket identifier

IPAddress (string) Client IP Address (“xxx.xxx.xxx.xxx”)

Port (integer) Client Port number

Checks if there are data available from specified client.

RETURN (boolean) : 0 = no bytes in queue

<>0 = number of bytes in queue

GetAddress(SocketId, IpAddress)

SocketId (integer) TCP Server socket identifier

IPAddress (integer - output) Server IP Address

Get port and address of the Server.

RETURN (integer) : The Server Port.

GetClientAddress(SocketId, ClientID, IpAddress)

SocketId (integer) TCP Server socket identifier

ClientId (integer) Index used to enumerate client connected to server.
Indicates the n-th client connected.
Range: 1to NumberOfClientConnected

IPAddress (integer – output) Server IP Address

Get the Port number and the IP Address the n-th client connected to the server.

RETURN (integer) : The Client Port.

NumberOfClientsConnected(SocketId, [Message])

SocketId (integer) TCP Server socket identifier

Message (array of bytes - optional) Binary message to send to the clients

Get the number of clients connected to the server.

RETURN (integer) : The number of clients connected to the server.

WriteString(SocketId, String, IPAddress, Port)

SocketId (integer) TCP Server socket identifier

String (string) String to send

IPAddress (string) Client IP Address ("xxx.xxx.xxx.xxx")

Port (integer) Client Port number

Send a characters string to the specified address and port client.

RETURN (integer) : The number of characters written.

WriteBuffer(SocketId, Buffer, NumBytes, IPAddress, Port)

SocketId (integer) TCP Server socket identifier

Buffer (array of bytes) Binary data to send

NumBytes (integer) Number of bytes to send

IPAddress (string) Client IP Address ("xxx.xxx.xxx.xxx")

Port (integer) Client Port number

Send binary data to the specified address and port client.

RETURN (integer) : The number of bytes written.

ReadString(SocketId, IPAddress, Port, String, NumChars)

SocketId (integer) TCP Server socket identifier

IPAddress (string – output) Client IP Address ("xxx.xxx.xxx.xxx")

Port (integer – output) Client Port number

String (string – output) Received string

NumChars (integer) Number of characters to read (0 = read all available)

Receive a characters string from a TCP Server socket.

RETURN (integer) : The number of characters read.

ReadStringClient(SocketId, IPAddress, Port, String, NumChars)

SocketId (integer) TCP Server socket identifier

IPAddress (string) Client IP Address (“xxx.xxx.xxx.xxx”)

Port (integer) Client Port number

String (string) Received string

NumChars (integer) Number of characters to read (0 = read all available)

Receive a characters string from a specified address and port client.

RETURN (integer) : The number of characters read.

ReadBuffer(SocketId, IPAddress, Port, Buffer, NumBytes)

SocketId (integer) TCP Server socket identifier

IPAddress (string – output) Client IP Address (“xxx.xxx.xxx.xxx”)

Port (integer – output) Client Port number

Buffer (array of bytes – output) Received data

NumBytes (integer) Number of bytes to read (0 = read all available)

Receive binary data from a TCP Server socket.

RETURN (integer) : The number of bytes read.

ReadBufferClient(SocketId, IPAddress, Port, Buffer, NumBytes)

SocketId (integer) TCP Server socket identifier

IPAddress (string) Client IP Address (“xxx.xxx.xxx.xxx”)

Port (integer) Client Port number

Buffer (array of bytes) Received data

NumBytes (integer) Number of bytes to read (0 = read all available)

Receive binary data from a specified address and port client.

RETURN (integer) : The number of bytes read.

Example 1

Send and receive data string:

```
serverIPAddress = "192.168.1.20"
serverIPPort = 20000
socketIDOpened = ESAHMI.ESAETH.TCPServer.OPEN( serverIPAddress, serverIPPort )

        result = ESAHMI.ESAETH.TCPServer.ISDATA( socketIDOpened, senderClientIPAddress, senderClientPort )
if result <> 0 then
        result = ESAHMI.ESAETH.TCPServer.READSTRING( socketIDOpened, senderClientIPAddress, senderClientPort,
                                                      stringReceived, 10 )
        ESAHMI.ESAmsgbox "Received " + CStr(result) + " character(s) from " + senderClientIPAddress + ":" +
        CStr(senderClientPort) + " " + stringReceived
End if

numOfClientConnected =
ESAHMI.ESAETH.TCPServer.NUMBEROFCLIENTSCONNECTED( socketIDOpened )
if numOfClientConnected > 0 then portClient = ESAHMI.ESAETH.TCPServer.GETCLIENTADDRESS( socketIDOpened,
                                                numOfClientConnected, ipAddressClient )
        result = ESAHMI.ESAETH.TCPServer.WRITESTRING( socketIDOpened, "example", ipAddressClient, portClient )
ESAHMI.ESAmsgbox "Written " + CStr(result) + " character(s) to " + ipAddressClient + ":" + CStr(portClient)
else
ESAHMI.ESAmsgbox "No client connected"
```

```
end if
ESAHMI.ESAETH.TCPServer.CLOSE socketIDOpened
```

Example 2

send and receive array of bytes:

```
serverIPAddress = "192.168.1.120"
serverIPPort = 20000
socketIDOpened = ESAHMI.ESAETH.TCPServer.OPEN( serverIPAddress, serverIPPort )
result = ESAHMI.ESAETH.TCPServer.ISDATA( socketIDOpened, senderClientIPAddress, senderClientPort )
if result <> 0 then
        Dim dataReceived
        bytesRead = ESAHMI.ESAETH.TCPServer.READBUFFER( socketIDOpened, senderClientIPAddress,
        senderClientPort, dataReceived, 10 )
For i = LBound(dataReceived) To UBound(dataReceived)
ESAHMI.ESAMsgBox "Bytes read (" & CStr(bytesRead) & ") from " & CStr(senderClientIPAddress) & ":" &
                 CStr(senderClientPort) & VBCrLf & dataReceived(i)
Next
End If
'prepare array of bytes
Dim message
ReDim message(0)
```

```
'send the byte 0 (zero) to the clients to get the number of the active 'connections
message(0) = CByte(0)
numOfClientConnected =
ESAHMI.ESAETH.TCPServer.NUMBEROFCLIENTSCONNECTED( socketIDOpened, message )
if numOfClientConnected > 0 then
portClient = ESAHMI.ESAETH.TCPServer.GETCLIENTADDRESS( socketIDOpened, numOfClientConnected,
                                                        ipAddressClient )
'create an array of 11 bytes
Dim dataToSend
ReDim dataToSend(10)
' initialize array of bytes
For kk = LBound(dataToSend) To UBound(dataToSend)
dataToSend(kk) = CByte(kk)
Next
bytesWritten = ESAHMI.ESAETH.TCPServer.WRITEBUFFER( socketIDOpened, dataToSend, 10, ipAddressClient,
                                                     portClient )
ESAHMI.ESAmsgbox "Written " + CStr(result) + " byte(s) to " + ipAddressClient + ":" + CStr(portClient)
end if
ESAHMI.ESAETH.TCPServer.CLOSE socketIDOpened
```

ESAETH.UDP

UDP provides the management of UDP protocol.

Methods :

Open(IPAddress, Port)

IPAddress	(string)	IP Address ("xxx.xxx.xxx.xxx")
Port	(integer)	Port number

Open an UDP socket.

RETURN (integer): The UDP socket identifier.

Close(SocketId)

SocketId	(integer)	UDP socket identifier

Close the UDP socket.

IsData(SocketId)

SocketId	(integer)	UDP socket identifier

Checks if there are data available.

RETURN (boolean) : 0 = no bytes in queue
<>0 = number of bytes in queue

GetAddress(SocketId, IpAddress)

SocketId	(integer)	UDP socket identifier
IPAddress	(integer - output)	IP Address

Get port and address of the specified UDP socket.

RETURN (integer) : The Port number.

WriteString(SocketId, String, IPAddress, Port)

SocketId (integer) UDP socket identifier

String (string) String to send

IPAddress (string) IP Address (“xxx.xxx.xxx.xxx”)

Port (integer) Port number

Send a characters string to specified address and port.

RETURN (integer) : The number of characters written.

WriteBuffer(SocketId, Buffer, NumBytes, IPAddress, Port)

SocketId	(integer)	UDP socket identifier
Buffer	(array of bytes)	Binary data to send
NumBytes	(integer)	Number of bytes to send
IPAddress	(string)	IP Address (“xxx.xxx.xxx.xxx”)
Port	(integer)	Port number

Send binary data to specified address and port.

RETURN (integer) : The number of bytes written.

ReadString(SocketId, IPAddress, Port, String, NumChars)

SocketId	(integer)	UDP socket identifier
IPAddress	(string - output)	IP Address (“xxx.xxx.xxx.xxx”)
Port	(integer - output)	Port number
String	(string - output)	Received string
NumChars	(integer)	Number of characters to read (0 = read all available)

Receive a characters string from an UDP socket.

RETURN (integer) : The number of characters read.

ReadBuffer(SocketId, IPAddress, Port, Buffer, NumBytes)

SocketId	(integer)	UDP socket identifier
IPAddress	(string - output)	IP Address (“xxx.xxx.xxx.xxx”)
Port	(integer - output)	Port number

Buffer	(array of bytes - output)	Received data
NumBytes	(integer)	Number of bytes to read (0 = read all available)

Receive binary data from an UDP socket.

RETURN (integer) : The number of bytes read.

Example 1

Send and receive data string:

```
socketIDOpened = ESAHMI.ESAETH.UDP.OPEN("192.168.1.20", 10000)
result = ESAHMI.ESAETH.UDP.ISDATA(socketIDOpened)
if result <> 0 then
result = ESAHMI.ESAETH.UDP.READSTRING( socketIDOpened,
IPSenderAddress,
IPSenderPort,
stringReceived,
10 )
ESAHMI.ESAmsgbox "Received " + CStr(result) + " character(s) from " +
CStr(IPSenderAddress) + ":" + CStr(IPSenderPort) + " " +
stringReceived
end if
result = ESAHMI.ESAETH.UDP.WRITESTRING( socketIDOpened,
"example",
"192.168.1.255",
20000 )
ESAHMI.ESAmsgbox "Written " + CStr(result) + " character(s)"
ESAHMI.ESAETH.UDP.CLOSE socketIDOpened
```

Example 2

Send and receive an array of bytes:

```
socketIDOpened = ESAHMI.ESAETH.UDP.OPEN("192.168.1.20", 10000)
result = ESAHMI.ESAETH.UDP.ISDATA(socketIDOpened)
if result <> 0 then
Dim dataReceived
bytesRead = ESAHMI.ESAETH.UDP.READBUFFER( socketIDOpened, IPSenderAddress, IPSenderPort, dataReceived,
                                                                10 )
For i = LBound(dataReceived) To UBound(dataReceived)
ESAHMI.ESAMsgBox "Bytes read (" & CStr(bytesRead) & ") from " & CStr(SenderIP) & ":" & CStr(SenderPort) & VBCrLf &
                                    dataReceived(i)
Next
end if
'create an array of 6 bytes
Dim dataToSend
ReDim dataToSend(5)

'initialize array of bytes
For kk = LBound(dataToSend) To UBound(dataToSend)
dataToSend(kk) = CByte(kk)
Next
bytesWritten = ESAHMI.ESAETH.UDP.WRITEBUFFER( socketIDOpened, dataToSend, 10, "192.168.1.255", 20000 )
```

```
ESAHMI.ESAmsgbox "Written " + CStr(bytesWritten) + " byte(s)"
ESAHMI.ESAETH.UDP.CLOSE socketIDOpened
```

ESAFILE

ESAFILE provides the access to the file system. It makes possible to create and remove files and folders. It makes also possible reading and writing on binary and text files, both ASCII and Unicode.

Methods - ESAFILE

Methods :

Copy(from_pathname, to_pathname)

From_pathname	(string)	Full path source file
To_pathname	(string)	Full path destination file

Copies an existing file to a new file.

Delete(pathname)

pathname	(string)	Full path file name

Deletes the specified file.

Rename(old_pathname, new_pathname)

old_pathname	(string)	Full path old file name
new_pathname	(string)	Full path new file name

Renames an existing file.

Exists(pathname)

pathname	(string)	Full path file name

RETURN (boolean) : TRUE = the file exists
FALSE = the file does NOT exist

IsDirectory(pathname)

pathname	(string)	Full path file name

Note: the specified path name must exist.

RETURN (boolean) : TRUE = the path name specifies a directory
FALSE = the path name does NOT specify a directory

GetFileLen(pathname)

pathname	(string)	Full path file name

RETURN (integer) : the size of the file (bytes)

SetFileLen(pathname)

pathname	(string)	Full path file name

Modifies the size of an existing file or create

AvailableSpace(pathname)

pathname	(string)	Full path directory name

RETURN (integer) : the amount of free space on the storage unit
where the specified directory resides (bytes)

MD(pathname)

pathname	(string)	Full path directory name

Creates a file system directory

RD(pathname)

pathname	(string)	Full path directory name

Deletes an existing empty directory

FindFirst(pathname)

pathname	(string)	Full path name, which can include wildcard characters, for example, an asterisk (*) or a question mark (?)

Searches a directory for a file or subdirectory with a name that matches a specific name
RETURN (string) : the name of the file or subdirectory found.
an empty string if no file found.

Example :

```
v = ESAHMI.ESAFILE.FindFirst("D:\test\*.*")

v = ESAHMI.ESAFILE.FindFirst("D:\test\*.txt")

v = ESAHMI.ESAFILE.FindFirst("D:\test\esa.txt")
```

FindNext()

Continues a file search from a previous call to FindFirst

RETURN (string) : the name of the file or subdirectory found.
an empty string if no more files found.

Example :

```
v = ESAHMI.ESAFILE.FindFirst("D:\test\*.*")
Do While v <> ""
   ESAHMI.ESAmsgbox v
   v = ESAHMI.ESAFILE.FindNext
Loop
```

Stream Methods - ESAFILE

Open(pathname, mode)

pathname	(string)	Full file path name
mode	(string)	Opening mode :

"r" Opens for reading.
If the file does not exist or cannot be found, the call fails.

"w" Opens an empty file for writing.
If the given file exists, its contents are destroyed.

"a" Opens for writing at the end of the file (appending);
creates the file first if it doesn't exist.

"r+" Opens for both reading and writing.
The file must exist.

"w+" Opens an empty file for both reading and writing.
If the given file exists, its contents are destroyed.

"a+" Opens for reading and appending;
the appending operation includes the removal of the EOF marker
before new data is written to the file and the EOF marker is restored
after writing is complete;
creates the file first if it doesn't exist.

In addition to the above values, "u" can be included to specify Unicode file (example: "r+u")

Opens a file for reading/writing.

Up to 32 files can be opened at a time.

Example:

```
file = "D:\note.txt"
ESAHMI.ESAFILE.Open file,"r+"
b = ESAHMI.ESAFILE.ReadByte(file)
ESAHMI.ESAmsgbox chr(b)
ESAHMI.ESAFILE.WriteByte file,asc("a")
ESAHMI.ESAFILE.Close file
```

Close(pathname)

Pathname	(string)	Full file path name

Closes a file.
The file must be previously opened with the Open method.

Rewind(pathname)

Pathname (string) Full file path name

Repositions the file pointer to the beginning of the file.
The file must be previously opened with the Open method.

Commit(pathname)

Pathname	(string)	Full file path name

Flushes a stream: writes to the file the contents of the buffer associated.
The file must be previously opened with the Open method.
See also the FileFlush property.

IsEOF(pathname)

Pathname	(string)	Full path file name

Tests for end-of-file on a stream.

Checks if a read operation attempted to read past the end of the file.

The file must be previously opened with the Open method.

RETURN (boolean) : TRUE = the current position is the end of the file

Example:

```
f1 = "C:\file.txt"
ESAHMI.ESAFILE.Open f1,"r"
flag = true
Do While flag
        v = ESAHMI.ESAFILE.ReadLine(f1,100)
        ESAHMI.ESAmsgbox v
        if ESAHMI.ESAFILE.IsEOF(f1) then
flag = false
end if
Loop
ESAHMI.ESAFILE.Close f1
```

GetLen(pathname)

Pathname	(string)	Full path file name

Gets the current length of an opened file.
The stream is flushed before calculating.
The file must be previously opened with the Open method.

RETURN (integer) : the current length of the file

SetUnicode(pathname)

Pathname	(string)	Full path file name

Writes the FFFEh unicode header.
The file must be previously opened with the Open method.

Example :

```
ESAHMIESAFILE.Open file,"w+u"
ESAHMIESAFILE.SetUnicode file
ESAHMIESAFILE.WriteByte file,asc("a")
```

SkipUnicode(pathname)

Pathname	(string)	Full path file name

Skips the FFFEh unicode header.

The file must be previously opened with the Open method.

Example :

```
ESAHMI.ESAFILE.Open file,"r+u"
ESAHMI.ESAFILE.SkipUnicode file
b = ESAHMI.ESAFILE.ReadByte(file)
```

WriteByte(pathname, byte)

Pathname	(string)	Full path file name
Byte	(integer)	Byte to write [0..255]

Writes one byte on a file.

The stream is flushed according to the FileFlush property.

The file must be previously opened with the Open method.

Example :

```
ESAHMI.ESAFILE.WriteByte file,asc("a")
ESAHMI.ESAFILE.WriteByte file,10
```

ReadByte(pathname)

Pathname	(string)	Full path file name

Reads one byte from a file.
The stream is flushed before reading.
The file must be previously opened with the Open method.

RETURN (integer) : the read byte

Example :

```
b = ESAHMI.ESAFILE.ReadByte(file)
```

WriteStr(pathname, str)

Pathname	(string)	Full path file name
str	(string)	String to write

Writes a string on a file.

The stream is flushed according to the FileFlush property.

The file must be previously opened with the Open method.

Example :

ESAHMI.ESAFILE.WriteStr file,"abcd"

ReadStr(pathname, count)

Pathname	(string)	Full path file name
count	(integer)	Maximum number of characters to be read

Reads a string from a text file. The number of characters actually read may be less than count if an error occurs or if the end of the file is encountered before reaching count.

The stream is flushed before reading.

The file must be previously opened with the Open method.

RETURN (string) : the read string

Example :

s = **ESAHMI.ESAFILE.ReadStr**(file,10)

WriteStrIdx(pathname, offset, str)

Pathname	(string)	Full path file name
Offset	(integer)	File offset [0...]
str	(string)	String to write

Writes a string on a file at specified offset.

The stream is flushed according to the FileFlush property.
The file must be previously opened with the Open method.

Example :

ESAHMI.ESAFILE.WriteStrIdx file,100,"abcd"

ReadStrIdx(pathname, offset, count)

Pathname	(string)	Full path file name
Offset	(integer)	File offset [0...]
count	(integer)	Maximum number of characters to be read

Reads a string from a text file at specified offset. The number of characters actually read may be less than count if an error occurs or if the end of the file is encountered before reaching count.

The stream is flushed before reading.
The file must be previously opened with the Open method.

Example :

s = **ESAHMI.ESAFILE.ReadStrIdx**(file,100,10)

ReadLine(pathname, count)

Pathname	(string)	Full path file name
count	(integer)	Maximum number of characters to be read

Reads a line from a text file. Reads characters from the current stream position to the first newline character, to the end of the stream, or until the number of characters read is equal to count, whichever comes first. The number of characters actually read may be less than count. The newline character, if read, is not included in the string.

The stream is flushed before reading.

The file must be previously opened with the Open method.

Example:

```
flag = true
Do While flag
    v = ESAHMI.ESAFILE.ReadLine(file,100)
    ESAHMI.ESAmsgbox v
    if ESAHMI.ESAFILE.IsEOF(file) then
        ESAHMI.ESAmsgbox "End of file!"
        flag = false
    end if
Loop
```

Properties - ESAFILE

Name	Type	Read-Write	Description
FileCount	integer	R	Number of currently opened files
FileFlush	boolean	RW	If TRUE, the file writing is direct. If FALSE, the file writing is buffered (default).

Notes :

The FileFlush property is set to FALSE by default, in order to prolong the life of the flash memory of the panel, minimizing the number of the write operations.

In any case, even if FileFlush is FALSE, the file stream is flushed when one of the following methods is called: Commit(), GetLen(), ReadByte(), ReadStr(), ReadLine(), ReadStrIdx(), WriteStrIdx(), Rewind(), Close().

ESAPAGEMGR

ESAPAGEMGR provides the access to the project pages and to some UI properties.

Methods - ESAPAGEMGR

ShowPage(Page)

Page	(string/ integer)	page name/ id

Shows the page specified.

Examples :

```
ESAHMI.ESAPAGEMGR.ShowPage "pagestart"
ESAHMI.ESAPAGEMGR.ShowPage 12
```

SetPageColor(Page, Color)

Page	(string/ integer)	page name/ id
Color	(integer)	RGB color

Changes the background color of the specified page.

GetPageColor(Page)

Page	(string/ integer)	page name/ id

RETURN (integer) : The background color of the specified page

GetPageWidth(Page)

Page (string/ integer) page name/ id

RETURN (integer) : The width of the specified page (pixel)

GetPageHeight(Page)

Page (string/ integer) page name/ id

RETURN (integer) : The height of the specified page (pixel)

ShowPageNext ( )
ShowPageNextFull( )
ShowPageNextPopup( )

Shows the next page.

ShowPagePrevious ( )
ShowPagePreviousFull ( )
ShowPagePreviousPopup ( )

Shows the previous page.

ShowPageLast( )

Shows the full screen page opened before the current one.

Keep a stack of 32 old pages. Only works with full screen pages.

ClosePopUp(Page)

Page	(string/ integer)	page name/ id

Closes a popup page.

ClosePopUpTop( )

Closes the top popup page.

ClosePopUpAll( )

Closes all the open popup pages.

ShowHelpPage(Page)

ShowHelpFullscreen( )

ShowHelpPopup( )

Page	(string/ integer)	page name/ id

Shows the help page.

CloseHelpPage(Page)

Page	(string/ integer)	page name/ id

Closes the help page.

CloseHelpPages( )

Closes all the opened help pages.

LanguageSet(Language)

LanguageNext( )

LanguagePrevious()

Language	(integer)	language identifier [1...]

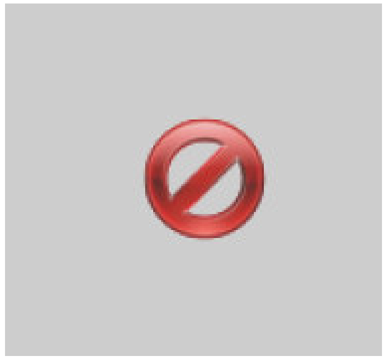
Change the current language.

LanguageGet( )

RETURN (integer) : The current language identifier [1...]

DisableInteraction(ShowSignal)

ShowSignal (boolean) if TRUE, an "Interaction Disabled" image is displayed



Disables any user interaction (touch-screen, mouse, keyboard, ...)

EnableInteraction()

Enables the user interaction (touch-screen, mouse, keyboard, ...)

IsPageOpen(Page)

Page	(string/ integer)	page name/ id

RETURN (boolean) : TRUE = the page is opened
FALSE = the page is NOT opened

GetPageName(PageID)

PageID (integer) Page Identifier

RETURN (string) : The name of the specified page

GetPageId(PageName)

PageName (string) Page name

RETURN (integer) : The identifier of the specified page

GetFullScreenName()

RETURN (string) : The name of the current full screen page

GetFullScreenId( )

RETURN (integer) : The identifier of the current full screen page

GetNumPopups( )

RETURN (integer) : The number of opened Popup pages

GetPopupName(index)

Index (integer) page index [0...]

RETURN (string) : The name of the specified popup page

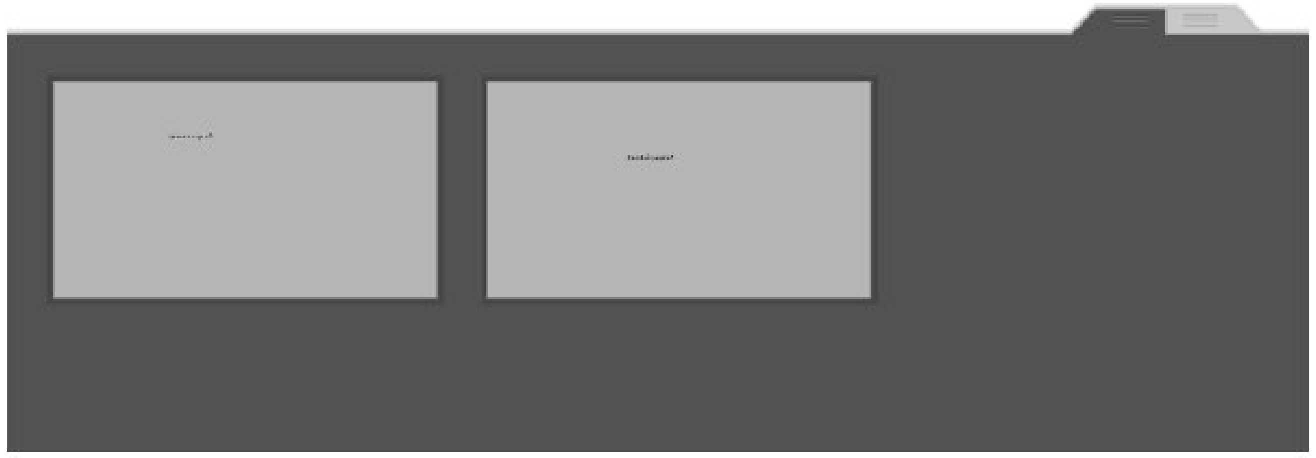
GetPopupId(index)

Index	(integer)	page index [0...]

RETURN (integer) : The identifier of the specified popup page

ShowRoadMap()

Displays the page Roadmap

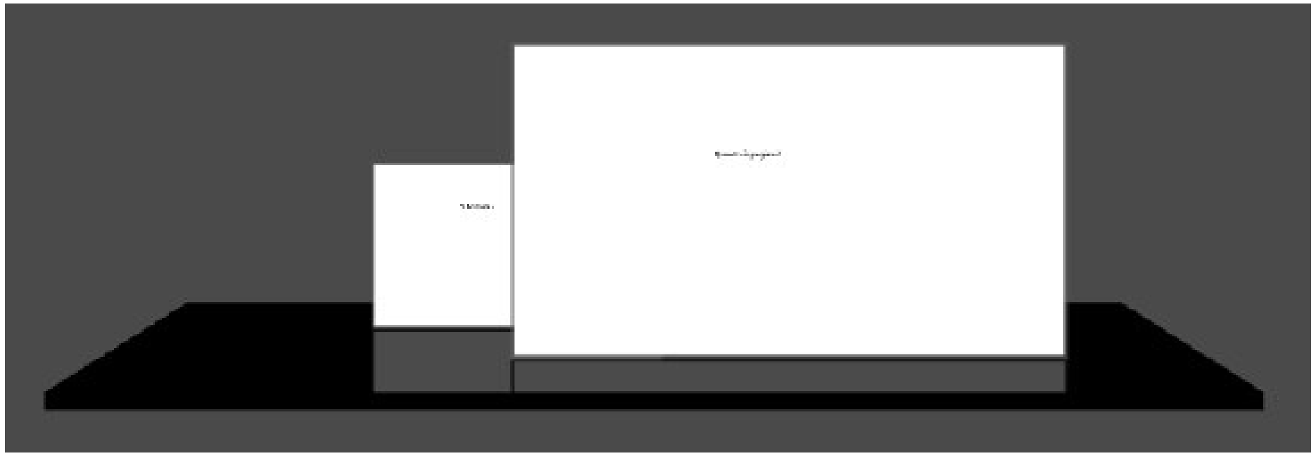


ShowPopupMap()

Displays the Popup Pages map

ShowSequenceRoll()

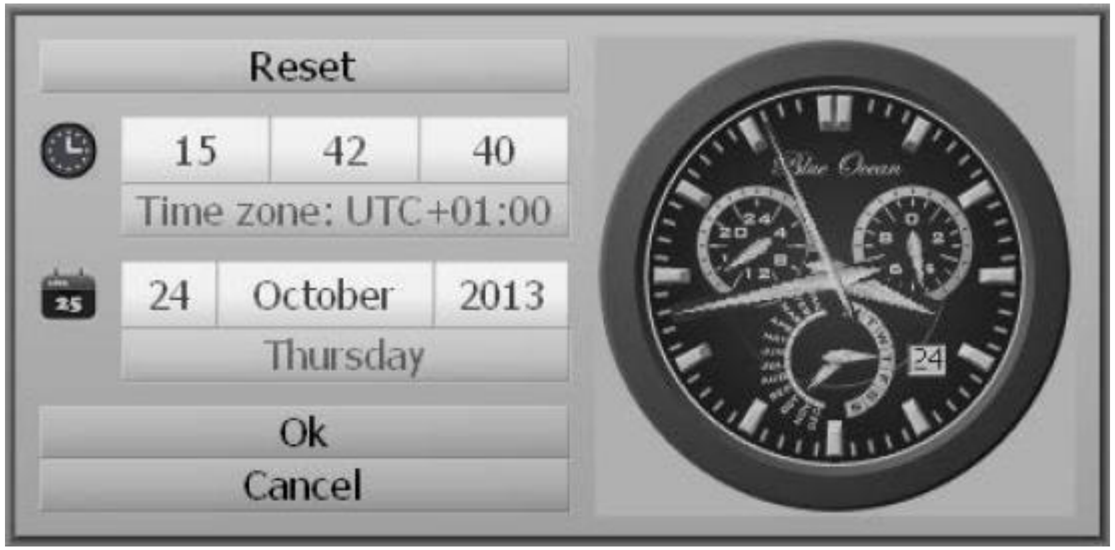
Displays the Pages Sequence Roll



ShowDateTimeBox([suspensive])

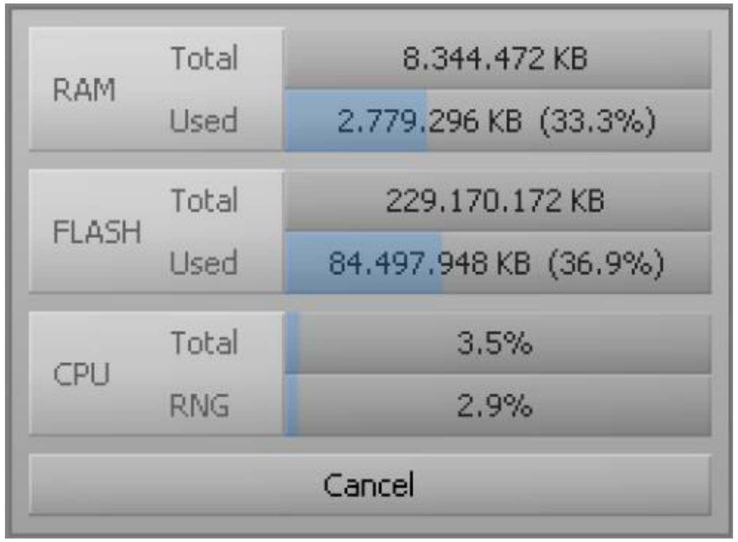
suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the Date/Time box



ShowResourceMonitorBox()

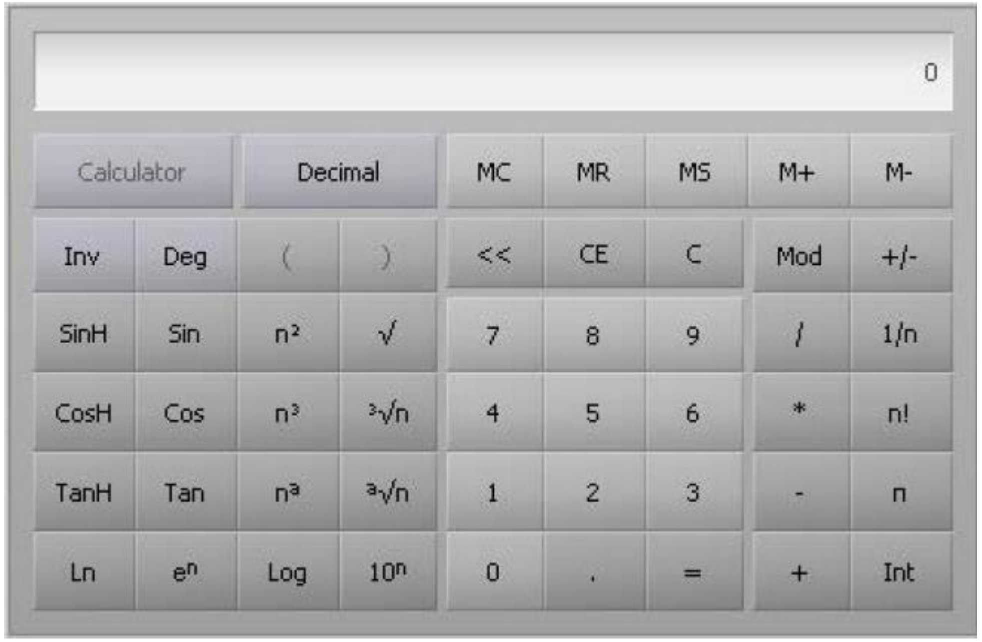
Displays a box showing some system data



RAM	Total	8.344.472 KB
	Used	2.779.296 KB (33.3%)
FLASH	Total	229.170.172 KB
	Used	84.497.948 KB (36.9%)
CPU	Total	3.5%
	RNG	2.9%

ShowCalculatorBox()

Displays the Calculator box



ESAPRN

ESAPRN provides the access to the printer.

Methods - ESAPRN

Start(UserFlag)

UserFlag	(boolean)	TRUE = show Print dialog box, FALSE = no dialog box

Starts a print session.

RETURN (boolean) : TRUE = Ok, FALSE = Cancel or Error

End()

Ends a print session.

Abort()

Terminates a print session.

NewPage()

Executes a form-feed.

WriteLN(Text)

Text	(string)	text to write

Writes a text and a CR-LF.

WriteRC(Row, Column, Text)

Row	(integer)	row (1..)
Column	(integer)	column (1..)
Text	(string)	text to write

Writes a text at specified row-column.

The column is correctly calculated with monospaced fonts only.

WriteXY(x, y, Text)

x	(integer)	x-coord. (0...)
y	(integer)	y-coord. (0...)
Text	(string)	text to write

Writes a text at the specified coordinates.

PrintImage(PathName, x, y, [width], [height])

PathName	(string)	Image file to print (full path)
x	(integer)	x-coord. (0...)
y	(integer)	y-coord. (0...)
width	(integer-optional)	image width
height	(integer-optional)	image height

Notes:
- If PathName specifies a filename only, the file will be loaded from the project images default folder.
- BMP, JPG, GIF and PNG images are supported.

- If Width or Height are not specified, the real image size is used.

Prints an image at the specified coordinates.

SetFont(Name, Size, [Bold], [Italic], [Underline], [Charset])

Name	(string)	typeface name of the font
Size	(integer)	font size in device unit
Bold	(boolean-optional)	TRUE = Bold (default=FALSE)
Italic	(boolean-optional)	TRUE = Italic (default=FALSE)
Underline	(boolean-optional)	TRUE = Underlined (default=FALSE)
Charset	(integer-optional)	0 = ANSI (default), 1= DEFAULT, 2 = SYMBOL

Set the print font.

Properties - ESAPRN

Name	Type	Read-Write	Description
FontSize	integer	RW	Font size in device unit (Courier New)
PageWidth	integer	R	Page width in pixels
PageHeight	integer	R	Page heigth in pixels
PageRows	Integer	R	Rows per page
PageColumns	integer	R	Page columns
MarginHor	integer	RW	Horizontal margin in pixels
MarginVert	integer	RW	Vertical margin in pixels

Example

```
set prn = ESAHMI.ESAPRN
go = prn.start(1)
if go <> 0 then
        prn.setfont "Courier New",20,1
        prn.writeln "Font Size = " &     prn.FontSize
        prn.writeln "Page Width = " & prn.PageWidth
        prn.writeln "Page Height = " & prn.PageHeight
        prn.writeln "Rows = " &          prn.PageRows
        prn.writeln "Columns = " &       prn.PageColumns
        prn.writeln "Hor. Margin = " & prn.MarginHor
        prn.writeln "Vert. Margin = " & prn.MarginVert
        prn.newpage
        prn.writexy 100,100,"100,100"
        prn.writerc 5,10,"005,010"
        prn.printimage "logo.png", 200, 200
        prn.printimage "\hard disk\temp\logo.bmp", 300, 300, 100, 200
        prn.end
end if
```

ESARECIPEMGR

ESARECIPEMGR provides the access to the project recipes.

Methods - ESARECIPEMGR

LoadRecipe(StructureName, RecipeName)

StructureName	(string)	structure name
RecipeName	(string)	recipe name

Transfers a recipe from the Archive to the Buffer Tags.

SaveRecipe(StructureName)

StructureName	(string)	structure name

Transfers a recipe from the Buffer Tags to the Archive.

Uses the Recipe-Name-buffer-tag.

SaveRecipeAs(StructureName, RecipeName)

StructureName	(string)	structure name
RecipeName	(string)	recipe name

Transfers a recipe from the Buffer Tags to the Archive with the specified name.

DeleteRecipe(StructureName, RecipeName)

StructureName	(string)	structure name
RecipeName	(string)	recipe name

Deletes the specified recipe from the Archive.

The recipe in the Archive is cleared; the corresponding record can be reused.

Call PackArchive() to recover the archive space.

DeleteAllRecipes(StructureName)

StructureName	(string)	structure name

Deletes all the recipes of the specified structure from the Archive.

The recipes in the Archive are cleared; the corresponding records can be reused.

A subseguent call to PackArchive() is recommended.

RenameRecipe(StructureName, OldRecipeName, NewRecipeName)

StructureName	(string)	structure name
RecipeName	(string)	recipe name

Renames in the Archive an existing recipe.

PackArchive(StructureName)

StructureName	(string)	structure name

Compacts the Archive of the specified structure.
The empty records are physically removed.

ClearTagBuffer(StructureName)

StructureName	(string)	structure name

Clear all the Buffer Tags of the specified structure.

RecipeDownload(StructureName, RecipeName, Synch)

StructureName	(string)	structure name
RecipeName	(string)	recipe name
Synch	(boolean)	TRUE = synchronized download

Transfers a recipe from the Archive to the Device Tags.

RecipeUpload(StructureName, RecipeName, Synch)

StructureName	(string)	structure name
RecipeName	(string)	recipe name
Synch	(boolean)	TRUE = synchronized upload

Transfers a recipe from the Device Tags to the Archive.

RecipeBufferDownload(StructureName, Synch)

StructureName	(string)	structure name
Synch	(boolean)	TRUE = synchronized download

Transfers a recipe from the Buffer Tags to the Device Tags.

RecipeBufferUpload(StructureName, Synch)

StructureName	(string)	structure name
Synch	(boolean)	TRUE = synchronized upload

Transfers a recipe from the Device Tags to the Buffer Tags.

GetRecipeCount(StructureName)

StructureName (string) structure name

RETURN (integer) : The number of valid recipes in the archive

GetRecipeRecords(StructureName)

StructureName (string) structure name
RETURN (integer) : The number of recipe-records in the archive, included the empty records.

Example:

```
set rm = ESAHMI.ESARECIPEMGR
snam = "Structure1"
num1= rm.GetRecipeCount(snam)
num2 = rm.GetRecipeRecords(snam)
s = num1 & " recipes, " & num2 & " records" & CHR(13) & CHR(13)
for i=1 to num2
v = rm.GetRecipeName(snam,i)
s = s & i & ": " & v & CHR(13)
next
ESAHMI.ESAmsgbox s
```

GetRecipeName(StructureName, RecipeId)

StructureName	(string)	structure name
RecipeId	(integer)	recipe id (record id) [1...]

RETURN (string) : The name of the recipe with specified numeric identifier.

GetTagName(StructureName, FieldName, Device)

StructureName	(string)	structure name
FieldName	(string)	field name ("RecipeName" and "Comment" also allowed)

Device (boolean) TRUE = device-tag, FALSE = buffer-tag

RETURN (string) : The name of the Tag associated to the specified data field.

RecipeExists(StructureName, RecipeName)

StructureName	(string)	structure name
RecipeName	(string)	recipe name

RETURN (boolean) : TRUE = recipe name found in the archive
FALSE = recipe name NOT found in the archive

RecipeExport(PathName, StructureName)
RecipeExportLocal(PathName, StructureName)

PathName	(string)	Full path export file name. If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed (Local version only).
StructureName	(string)	structure name

Exports the recipes of the specified structure into an ESA-format CSV file.

RETURN (integer) : The number of exported recipes.

RecipeExportAll(PathName)

PathName	(string)	Full path export file name

Exports all the recipes of all the structures into an ESA-format CSV file.
RETURN (integer) : The number of exported recipes.

RecipeExportAllLocal(PathName, [suspensive])

PathName	(string)	Full path export file name. If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive (boolean-optional) TRUE = suspensive dialog box

FALSE = non-suspensive dialog box

Exports all the recipes of all the structures into an ESA-format CSV file.

RETURN (integer) : The number of exported recipes.

RecipeImport(PathName, StructureName, [suspensive])

PathName	(string)	Full path import file name. If PathName is an empty string (“”), a dialog box that allows the user to select a path is displayed.
StructureName	(string)	structure name

suspensive	(boolean-optional)	TRUE = suspensive dialog box
		FALSE = non-suspensive dialog box

Imports the recipes of the specified structure from a file.

File formats supported:
- ESA-format CSV, (Unicode)
- Standard CSV, semicolon separated fields (Unicode and ANSI)
- Standard TXT, TAB separated fields (Unicode and ANSI)

The ImportedNew and ImportedOld properties are set.

RETURN (integer) : The number of imported recipes.

RecipeImportAll(PathName, [suspensive])

PathName	(string)	Full path import file name If PathName is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive	(boolean-optional)	TRUE = suspensive dialog box
		FALSE = non-suspensive dialog box

Imports all the recipes of all the structures from an ESA-format CSV file.
The ImportedNew and ImportedOld properties are set.

RETURN (integer) : The number of imported recipes.

Example:

```
a = ESAHMI.ESARECIPEMGR.RecipeImportAll(“c:\import.csv”)
b = ESAHMI.ESARECIPEMGR.ImportedNew
c = ESAHMI.ESARECIPEMGR.ImportedOld
s = "Imported:" & a & " New:" & b & " Replaced:" & c
ESAHMI.ESAmsgbox s
```

RecipeCompare(StructureName, RecipeName1, RecipeName2)

StructureName	(string)	structure name
RecipeName1	(string)	recipe name 1
RecipeName2	(string)	recipe name 2

Compares the field values of two recipes with same structure.

RETURN (boolean) : TRUE = recipes identical
FALSE = recipes different

RecipeLoadBox(StructureName, [suspensive])

StructureName (string) structure name
suspensive (boolean-optional) TRUE = suspensive,
FALSE = non-suspensive
Displays the Recipe Load box in order to load a recipe.

RecipeSaveBox(StructureName, [suspensive])

StructureName (string) structure name
suspensive (boolean-optional) TRUE = suspensive,

FALSE = non-suspensive

Displays the Recipe Save box in order to save a recipe.

RecipeSaveAsBox(StructureName, [suspensive])

StructureName	(string)	structure name
suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the Recipe Save As box in order to save a recipe.

RecipeDeleteBox(StructureName, [suspensive])

StructureName	(string)	structure name
suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the Recipe Delete box in order to delete a recipe.

RecipeRenameBox(StructureName, [suspensive])

StructureName	(string)	structure name
suspensive	(boolean-optional)	TRUE = suspensive,

FALSE = non-suspensive

Displays the Recipe Rename box in order to rename a recipe.

RecipeDownloadBox(StructureName, Synch, [suspensive])

StructureName	(string)	structure name
Synch	(boolean)	TRUE = synchronized download
suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the Recipe Download box in order to download a recipe.

RecipePrint(StructureName)
RecipePrintLocal(StructureName)

StructureName	(string)	structure name

Prints the recipes of the specified structure.

RETURN (integer) : The number of printed recipes.

RecipePrintAll()
RecipePrintAllLocal()

Prints all the recipes of all the structures.

RETURN (integer) : The number of printed recipes.

Properties - ESARECIPEMGR

Name	**Type**	**Read-Write**	**Description**
Busy	integer	R	0 = no transfer in progress 1 = transfer in progress
ImportedNew	Integer	R	the number of new imported recipes after a *RecipeImport()* call
ImportedOld	Integer	R	the number of replaced recipes after a *RecipeImport()* call

ESASAMPLEMGR

ESASAMPLEMGR provides the access to the project samples.

Methods - ESASAMPLEMGR

Enable(SampleName)

SampleName	(string)	Sample name

Enables the sampling activity.

Disable(SampleName)

SampleName	(string)	Sample name

Disables the sampling activity.

ResetSamples(SampleName)

SampleName	(string)	Sample name

Removes all the samples from the sampling buffer.

AcquireSample(SampleName)

SampleName	(string)	Sample name

Requests a “one-shot” acquisition of a new sample from the source.

ExportSamples(PathName, SampleName)

Pathname	(string)	Full path file name
SampleName	(string)	Sample name

Exports in a files all the samples of the sampling buffer.

ExportSamplesLocal(PathName, SampleName, [suspensive])

Pathname	(string)	Full path file name If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.
SampleName	(string)	Sample name

suspensive	(boolean-optional)	TRUE = suspensive dialog box
		FALSE = non-suspensive dialog box

Exports in a files all the samples of the sampling buffer.

ImportSamplesLocal(Pathname, PageName, ControlName, [suspensive])

Pathname (string)	Full path file name. If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.
PageName (string)	Page name.
ControlName (string)	Control name.

suspensive (boolean-optional)	TRUE = suspensive dialog box
	FALSE = non-suspensive dialog box

Imports tracking data from text file.

ExportInProgress(SampleName)
ExportInProgressLocal(SampleName)

SampleName (string)	Sample name

States whether there is an export operation in progress on the specified sampling buffer.
RETURN (boolean) : TRUE if an export operation is currently in progress.

WaitForExport(SampleName)
WaitForExportLocal(SampleName)

SampleName (string) Sample name

Wait the end of an export operation.

TerminateExport(SampleName)

TerminateExportLocal(SampleName)

SampleName (string) Sample name

Stops an export operation.

FlushPersistentData(SampleName)

SampleName (string) Sample name

Saves on disk the collected persistent samples.

SamplesPrint(SampleName)

SamplesPrintLocal(SampleName)

SampleName (string) Sample name

Prints the sample data.

ESATAG

ESATAG provides the access to the project TAGs. Moreover, it makes possible to access directly the external device.

Methods - ESATAG

GetTagId(TagName)

TagName (string) The name of the Tag

RETURN (integer) : The Tag identifier

GetTagName(TagId)

TagId (integer) The Tag identifier

RETURN (string) : The Tag name

GetTagValueType(TagName)

TagName (string) The name of the Tag

RETURN (integer) : The type of the Tag value :

16	1- byte signed integer	(I1)
2	2-byte signed integer	(I2)
3	4-byte signed integer	(I4)
17	1-byte unsigned integer	(UI1)
18	2-byte unsigned integer	(UI2)
19	4-byte unsigned integer	(UI4)
4	4-byte floating point	(R4)
5	8-byte floating point	(R8)
11	boolean	(BOOL)
8	string	(BSTR)

Note:
Returned Tag Array types are defined by adding 8192 to the above values.
(ex. array of string: 8200 = 8 + 8192)

GetTagStrLength(TagName)

TagName	(string)	The name of the Tag

RETURN (integer) : The length of a string Tag

GetTagArraySize(TagName)

TagName	(string)	The name of the Tag

RETURN (integer) : The number of elements of a Tag-array

GetDeviceId(DeviceName)

DeviceName	(string)	The name of the Device

RETURN (integer) : The Device identifier

GetDeviceName(DeviceId)

DeviceId	(integer)	The Device identifier

RETURN (string) : The Device name

GetCurrentValue(TagName)

TagName	(string)	The name of the Tag

Reads the value currenlty stored in the tag (doesn't access the value in the device)

RETURN (variant) : The Tag value

ReadValue(TagName)

TagName	(string)	The name of the Tag

Reads the value of a tag from the device

RETURN (variant) : The Tag value

Example:

```
v = ESAHMI.ESATAG.ReadValue("Tag")
```

WriteValue(TagName, Value)

TagName	(string)	The name of the Tag
Value	(variant)	New tag value

Writes a new tag value on the device

Example:

ESAHMI.ESATAG.WriteValue "Tag",123

ReadElement(TagName, Index)

TagName	(string)	The name of the Tag
Index	(integer)	Index of the needed element [0...]

Reads the value of a single element of a tag array from the device

RETURN (variant) : The element value

WriteElement(TagName, Index, Value)

TagName	(string)	The name of the Tag
Index	(integer)	Index of the needed element [0...]
Value	(variant)	New tag value

Writes the value of a single element of a tag array on the device

Example

```
a = Array(10,20,30)
ESAHMI.ESATAG.WriteValue "TagArray",a
r = ESAHMI.ESATAG.ReadElement( "TagArray",2 )
r = r + 1000
ESAHMI.ESATAG.WriteElement "TagArray",2,r
q = ESAHMI.ESATAG.ReadValue( "TagArray" )
b = q(1)
```

ReadBit(TagName, Index)

TagName	(string)	The name of the Tag
Index	(integer)	Index of the needed bit [0...]

Reads the value of a single bit of a numeric tag (or an array) from the device

RETURN (boolean) : The bit value

WriteBit(TagName, Index, Value)

TagName	(string)	The name of the Tag
Index	(integer)	Index of the needed element [0...]
Value	(boolean)	New tag value

Writes the value of a single bit of a numeric tag (or an array) on the device

ReadItem(DeviceId, AreaId, ValueType, StringLen, ArraySize, IsBCD, AF1, [AF2], [AF3], [AF4], [AF5], [AF6], [AF7], [AF8])

DeviceId	(integer) Device identifier
AreaId	(integer) Area identifier
ValueType	(integer) Value type (see GetTagValueType)
StringLen	(integer) String length (valid for String tag)
ArraySize	(integer) Number of array elements (valid for tag-array)
IsBCD	(boolean) TRUE = BCD coding
AF1	(variant) Address Field 1
AF2 – AF8	(variant-optional) Address Fields 2 . . 8

Reads a value directly from the device

RETURN (variant) : The device value

WriteItem(Value, DeviceId, AreaId, ValueType, StringLen, ArraySize, IsBCD, AF1, AF2, AF3, AF4, AF5, AF6, AF7, AF8)

Value	(variant) Value to write
DeviceId	(integer) Device identifier
AreaId	(integer) Area identifier
ValueType	(integer) Value type (see GetTagValueType)

StringLen	(integer) String length (valid for String tag)
ArraySize	(integer) Number of array elements (valid for tag-array)
IsBCD	(boolean) TRUE = BCD coding
AF1	(variant) Address Field 1
AF2 – AF8	(variant-optional) Address Fields 2 . . 8

Writes a value directly to the device

SetTagOffscan(TagName, offScan)

TagName	(string)	The name of the Tag
offScan	(boolean)	TRUE = set off-scan, FALSE = reset off-scan

Sets the Tag off-scan state

SetDeviceOffscan(DeviceName, offScan)

DeviceName	(string)	The name of the Device
offScan	(boolean)	TRUE = set off-scan, FALSE = reset off-scan

Sets the Device off-scan state

IsOffline(TagName)

TagName	(string)	The name of the Tag

Checks if the specified Tag is Off-Line.

RETURN (boolean) : TRUE = Tag Off-Line, FALSE = Tag On-Line

ESATIMER

ESATIMER provides the access to the projects Timers.

Methods - ESATIMER

Start(timername)

timername	(string)	timer to be started

Starts the specified timer.

Stop(timername)

timername	(string)	timer to be stopped

Stops the specified timer.

Suspend(timername)

timername (string) timer to be suspended

Suspends the specified timer.

SetTimerValue(timername, value)

timername	(string)	timer to be set
value	(integer)	limit value to set

Sets the limit of the specified timer.

GetTimerValue(timername)

timername	(string)	timer name

RETURN (integer) : The limit of the specified timer.

SetProgress(timername, value)

timername	(string)	timer to be set
value	(integer)	progress value to set

Sets the progress value of the specified timer.

GetProgress(timername)

timername	(string)	timer name

RETURN (integer) : The progress value of the specified timer.

IsStarted(timername)

timername	(string)	timer name

RETURN (boolean) : TRUE if the timer has been started.

IsSuspended(timername)

timername	(string)	timer name

RETURN (boolean) : TRUE if the timer has been suspended.

ESAUSERMGR

ESAUSERMGR provides the management of the projects Users.

Methods - ESAUSERMGR

Add(username, groupname, mode, password, [language], [email], [phone], [validity])

username	(string)	user name to add
groupname	(string)	group name
mode	(integer)	password mode (0 = alphanumeric, 1= graphic)
password	(string)	password
language	(integer-optional)	language identifier [1...] (0 = no language change)
email	(string-optional)	email address
phone	(string-optional)	phone number
validity	(integer-optional)	days of password validity (0 = no limit)

Adds a new user.

Remove(username)

username	(string)	user name to remove

Removes an existing user.

Login(username, password)

username	(string)	user name to log-in
password	(string)	password

Executes the login for an existing user.

Logout()

Executes the logout for the current user.

ChangePassword(username, mode, password)

username	(string)	user name
mode	(integer)	password mode (0 = alphanumeric, 1= graphic)
password	(string)	password

Changes the password of an existing user.

ChangePasswordValidity(username, validity)

username	(string)	user name
validity	(integer)	days of password validity (0 = no limit)

Changes the password validity of an existing user.

ChangeGroup(username, groupname)

username	(string)	user name
groupname	(string)	new group name

Changes the group for an existing user.

ChangeLanguage(username, language)

username	(string)	user name
language	(integer)	language identifier [1...] (0 = no language change)

Changes the language for an existing user.

ChangeEmail(username, email)

username	(string)	user name
email	(string)	email address

Changes the email address for an existing user.

ChangeTelNumber(username, phone)

username	(string)	user name
phone	(string)	phone number

Changes the phone number for an existing user.

UsersFlush()

Flushes on disk the user log file.

GetCurrentUserName()

RETURN (string) : The Name of the currently logged user.

GetCurrentVisibility()

RETURN (integer) : The Visibility Level of the currently logged user.

GetCurrentInteractivity()

RETURN (integer) : The Interactivity Level of the currently logged user.

GetCurrentGroup()

RETURN (string) : The Group name associated to the currently logged user.

GetUserVisibility(username)

username	(string)	user name

RETURN (integer) : The Visibility Level of the specified user.

GetUserInteractivity(username)

username	(string)	user name

RETURN (integer) : The Interactivity Level of the specified user.

GetUserGroup(username)

username	(string)	user name

RETURN (string) : The Group name associated to the specified user.

GetUserLanguage(username)

username	(string)	user name

RETURN (integer) : The language associated to the specified user [1...] (0 = no language change)

GetUserEmail(username)

username	(string)	user name

RETURN (string) : The email address associated to the specified user.

GetUserTelNumber(username)

username	(string)	user name

RETURN (string) : The phone number associated to the specified user.

GetUserPasswordValidity(username)

username	(string)	user name

RETURN (integer) : The days of password validity (0 = no limit).

LogExport(pathname)

Pathname	(string)	Full path file name.

Exports the Users Log to the specified file name.

RETURN (integer) : The number of records exported.

LogExportLocal(pathname, [suspensive])

Pathname	(string)	Full path file name. If Pathname is an empty string (“”), a dialog box that allows the user to select a path is displayed.

suspensive	(boolean-optional)	TRUE = suspensive dialog box
		FALSE = non-suspensive dialog box

Exports the Users Log to the specified file name.

RETURN (integer) : The number of records exported.

LoginBox([suspensive])

suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the User Login box to perform a user log-in.

LoginPasswordBox(username, [suspensive])

username	(string)	user name
suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the Password Login box to perform a log-in for the specified user.

AddBox([suspensive])

suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the User Add box to add a user to the project.

RemoveBox([suspensive])

suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the User Remove box to remove a user from the project.

ChangeInfoBox([suspensive])

suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Displays the User Change Info box to change a user setting (password, ...)

UserLock(username)

username	(string)	user name

Locks the specified user.

UserUnlock(username)

username	(string)	user name

Unlocks the specified user.

UsersPrint()
UsersPrintLocal()

Prints the Users Log.

RETURN (integer) : The number of printed events.

UserJoinList(username, mailinglist, type)

username	(string)	user name
mailinglist	(string)	mailing list
type	(integer)	recipient type (0=normal, 1=copy, 2=hidden)

Adds the user to a mailing list.

UserLeaveList(username, mailinglist)

username	(string)	user name
mailinglist	(string)	mailing list

Removes the user from a mailing list.

UserResetLists(username)

username	(string)	user name

Removes the user from all the mailing lists.

SendMailSingle(email, subject, message)

email	(string)	email address
subject	(string)	email subject
message	(string)	email message

Sends an email.

SendMailList(mailinglist, replyto, attachments, subject, message)

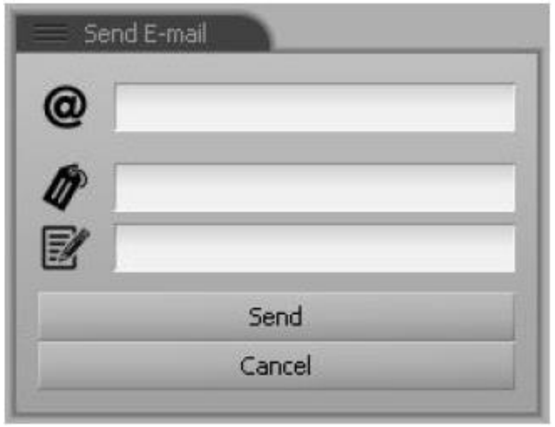
mailinglist	(string)	mailing list name
attachments	(string)	attachment path names TAB (CHR(9)) separated
subject	(string)	email subject
message	(string)	email message

Sends an email to a mailing list.

SendMailBox([suspensive])

suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Shows a “Send E-mail” box.



SendSmsSingle(phone, message)

phone	(string)	phone number
message	(string)	sms message

Sends an s.m.s.

SendSmsList(phonelist, message)

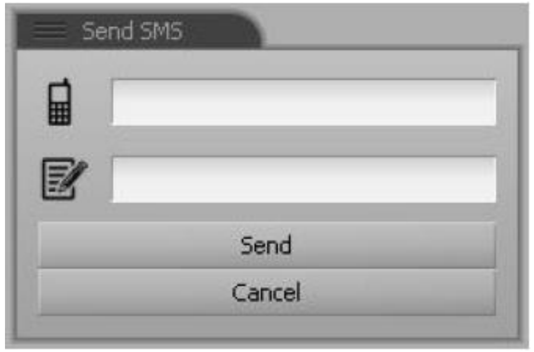
phonelist	(string)	phone numbers list name
message	(string)	sms message

Sends an s.m.s. to a list of numbers.

SendSmsBox([suspensive])

suspensive	(boolean-optional)	TRUE = suspensive, FALSE = non-suspensive

Shows a “Send SMS” box.



ImportNetworkUsers()

Imports the users list from the network specified in the project

Examples of using Scripts

Example 1- Analysis of variables and launching events

In this example we will suppose we have a project in which we configure a page, a variable, an alarm and the controls assigned to the page.

Using CREW we set the objects we need while running the Script. We set a variable, calling it ‘Tag’ (the names of the objects assigned using CREW are important as this is the key to accessing them using Scripts) of the Integer type assigning an initial value of 0.

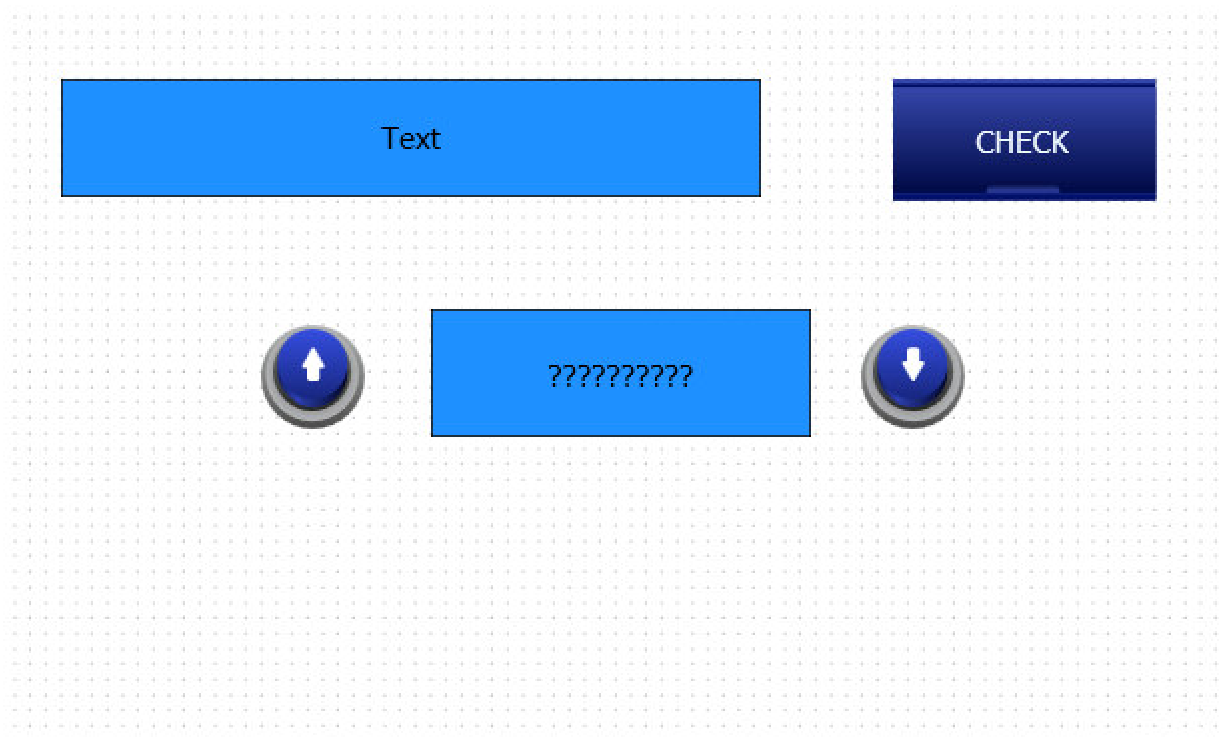
In addition, we set a generic alarm (‘Alarm’) that will be set off when the variable ‘Tag’ assumes the value 10. We remember to set in Alarms, in the "Alarm Signals" mask (see Alarm Signals section), the display of one of the available alarm signals.

We set a page called ‘Page’ in which we insert a label (called ‘Label’) and a touch button (‘Touch Button’) to which we assign the Script (‘Script) corresponding to the event ‘onReleased’.

Using Project Explorer, we drag the variable to the work area to create a dynamic field showing its value in runtime (useful for constantly monitoring its value).

We add two buttons to which we assign the predefined functions of increase-decrease value acting on the variable ‘Tag’ so as to be able to change the value in runtime.

The page created will look like this :



Our Script must be able to get the value of the variable ‘Tag’, check that the value is less than 5 and, should this not be the case, launch an alarm, edit the layout of the label and the page and take the variable to a low value.

To get the value of the variable, we use ESATAG and save it into variable ‘a’ with the following instruction :

```
a=ESAHMI.ESATAG.ReadValue ("Tag")
```

Now let us analyze the received value: if the value is greater than or equal to 5, an alarm is raised. Using CREW we set an alarm to be activated when the value 10 was reached, so we are certain that it is the Script activating it now.

The control and activation code uses the object ESAALARMMGR as indicated by the following rows :

```
If a>4 Then

ESAHMI.ESAALARMMGR.RaiseAlarm("Alarm")
End If
```

We can also run other instructions within the same condition such that when we change the value of the variable and launch the Script other changes will be applied, too.

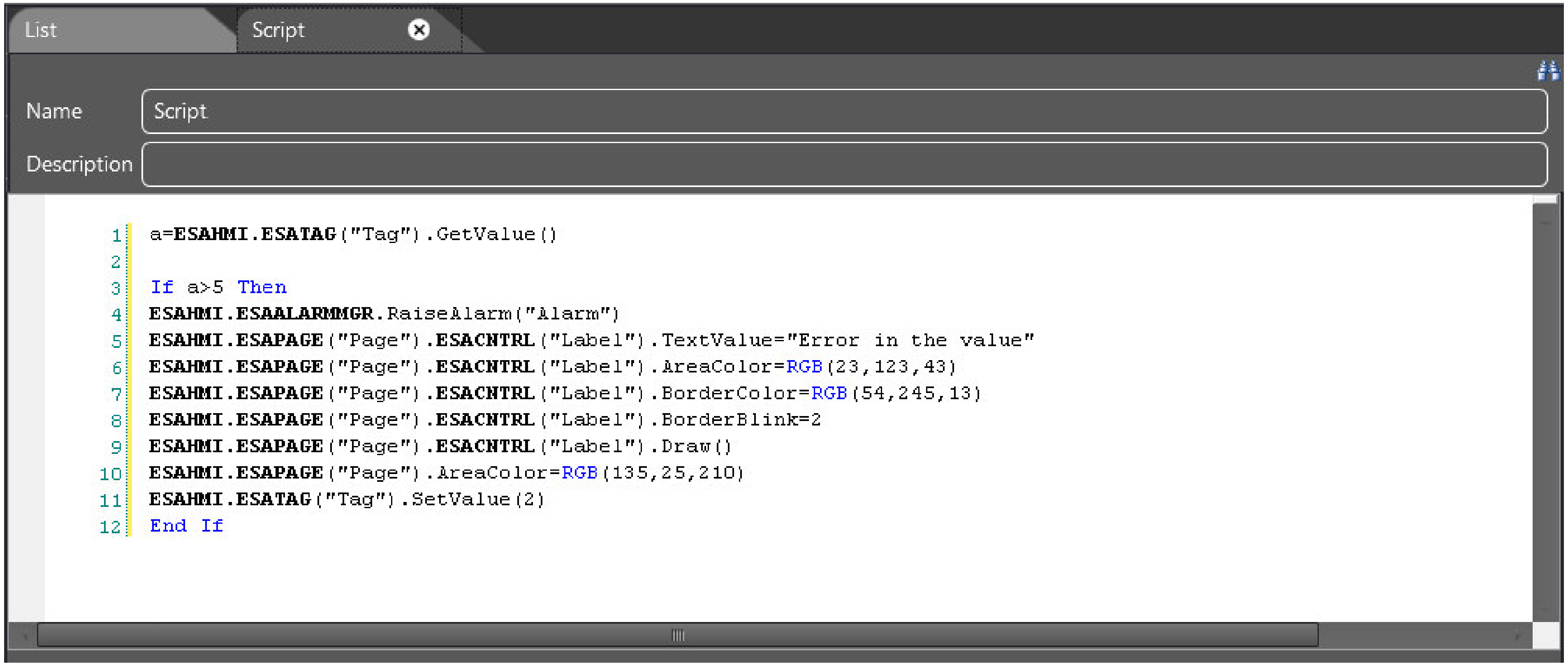
For example, we change the text, the color and the blinking of the label (object ESACNTRL, remembering to invoke the Draw method related to the label) and the background of the page (object ESAPAGE) as set out below :

```
If a>4 Then
ESAHMI.ESAALARMMGR.RaiseAlarm("Alarm")
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").TextValue="Errore nel valore"
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").AreaColor=RGB (23,123,43)
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderColor=RGB (54,245,13)
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").Border-Blink=2
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").Draw()
ESAHMI.ESAPAGE("Page").AreaColor=RGB(25,25,25)
End If
```

Finally, we re-establish an admissible value for the variable with the following instruction:

```
ESAHMI.ESATAG.WriteValue "Tag",2
```

The final code inserted in the CREW editor is the following:



```
a=ESAHMI.ESATAG("Tag").GetValue()

If a>5 Then
ESAHMI.ESAALARMMGR.RaiseAlarm("Alarm")
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").TextValue="Error in the value"
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").AreaColor=RGB(23,123,43)
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderColor=RGB(54,245,13)
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderBlink=2
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").Draw()
ESAHMI.ESAPAGE("Page").AreaColor=RGB(135,25,210)
ESAHMI.ESATAG("Tag").SetValue(2)
End If
```

Example 2 - Page access according to user level

Another example of using Scripts is the way access to project pages is managed according to the level of the user currently logged onto the terminal.

Using CREW we can set the objects we need while the Script is run; we set two levels of use (see "Groups" section), assigning a password for levels 3 and 8, for example.

Remember that when the project starts the predefined level is 10, that is, the lowest.

We add 3 buttons to the default page (‘Page’): one recalling the Script, the other two the log-in and log-out functions respectively.

Finally we set two new pages (‘Page_1’ and ‘Page_2’) that will be recalled by the Script depending on the user level.

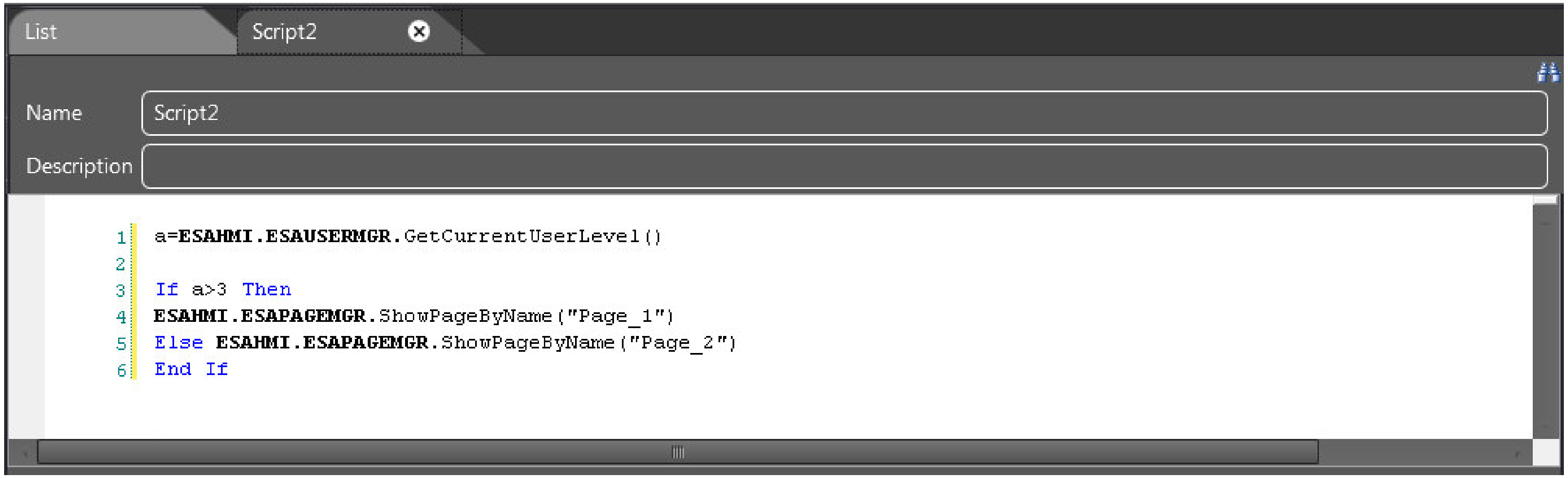
Let us look now at the implementation of the code: first of all we must use the object USERMGR to get the level of the user currently logged in :

```
a=ESAHMI.ESAUSERMGR.GetCurrentUserLevel()
```

Now we need merely create a check condition for this level (the function returns an integer). The credentials of the user will determine which page is displayed :

```
If a>3 Then
ESAHMI.ESAPAGEMGR.ShowPageByName("Page_1")
Else ESAHMI.ESAPAGEMGR.ShowPageByName("Page_2")
End If
```

The complete Script code is as follows:



Name: Script2

Description:

```
a=ESAHMI.ESAUSERMGR.GetCurrentUserLevel()

If a>3 Then
ESAHMI.ESAPAGEMGR.ShowPageByName("Page_1")
Else ESAHMI.ESAPAGEMGR.ShowPageByName("Page_2")
End If
```

Example 3 - Exporting alarms to a file chosen by the user

Another example of how CREW Scripts can be used is provided by the use of value fields to receive data to be used to invoke dynamic functions.

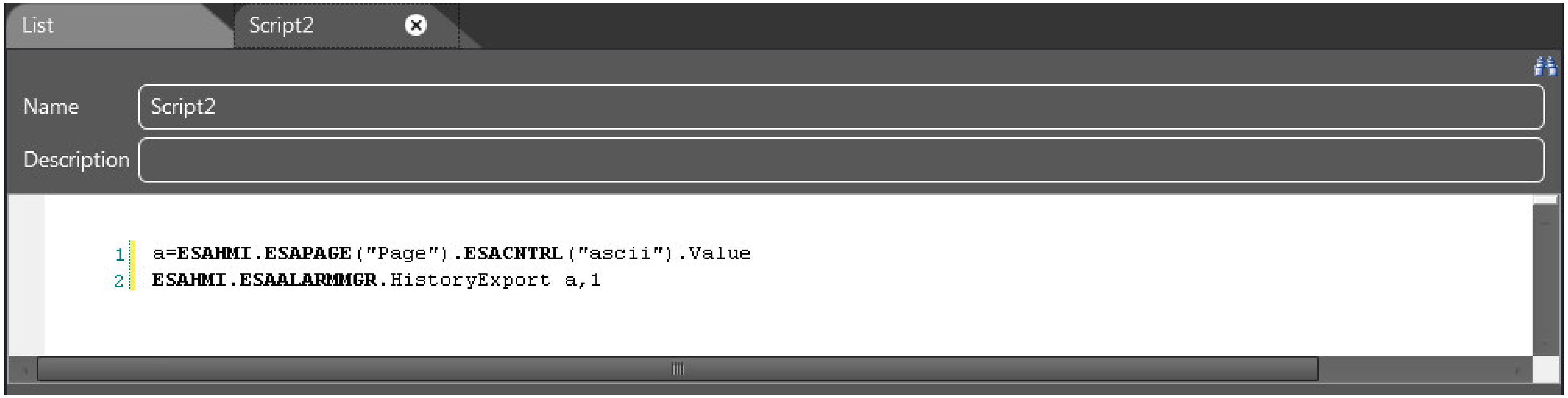
We insert a complex field into a page and this displays the Alarm history, an ASCII field ('ASCII', assigned to a string-variable) and a button to which we assign a Script (event onReleased).

The Script reads the value of the ASCII field and saves it in variable 'a' using the following instruction:

a=ESAHMI.ESAPAGE("Page").ESACNTRL("ascii").Value

Then we invoke the Export alarms function to which we pass the string that has just been read:

ESAHMI.ESAALARMMGR.HistoryExport a,1



```
a=ESAHMI.ESAPAGE("Page").ESACNTRL("ascii").Value
ESAHMI.ESAALARMMGR.HistoryExport a,1
```

Naturally this is only a simple example useful for illustrating the ease of programming via scripting that makes creating the project extremely dynamic.

Example 4 - Saving a Recipe into a memory

In this subsection we will show how it is possible, for example, using a Script to force the loading, the saving and exporting of certain recipes when a bit in the device is raised. To do this, we assign this Script to the event OnRawValueChange of the control bit (in our case, the variable 'Control'). The PLC raises the status of the bit every X minutes allowing the Script to operate. The Script operates the saving of the export file using a format of the type ric_DATA_ORA.xml.

In the project we create a type of recipe called 'Proportions' and define it as we wish; this will be used in the Script.

In this example we also introduce the use of a function that checks a variable and returns a value; namely, the values relating to the days, months, hours, minutes and seconds returned by the functions VBSCRIPT can be values of 1 digit. So that all files saved have the same format and the same length, we write a function of a few rows that adds a 0 in front of a digit if it is less than 10.

Using CREW we create a Script in the usual way, but in the general page we assign a name ('addzero'), a type of returned value (Variant) and an input value ('value', numeric).

We have created the structure of our function: now we write its code:

```
If value<10 Then
value="0" & value
End If
addzero=value
```

If the input value ('value') is less than ten (that is, consists of only one digit), add the string "0" to the variable and finally it returns the value of 'Value' (if the cycle is not accessed the function simply returns the value received as an input parameter).

The following is an example of applying this function :

addzero(5) is invoked by giving the value 5, and returns the value ‘05’.

Now let us analyze the code of our main Script :

```
a=ESAHMI.ESATAG("Check").GetRawValue()
```

First of all we read the raw value of the Check variable and if its value is 1 we run our operations (this way we avoid executing them when the bit passes from 1 to 0).

The If cycle is as follows:

```
If (a=1) Then
ESAHMI.ESARECIPEMGR.RecipeBufferUpload "Proportions",0
ESAHMI.ESARECIPEMGR.SaveRecipe "Proportions","Recipe",0
End If
```

We execute the upload of the recipe loaded onto the PLC (type is ‘Proportions’) in the first rows of the cycle, while in the second row we save that recipe (using the name ‘Recipe’) onto the terminal.

Now all we need is the save phase that is run using the following instruction :

```
ESAHMI.ESARECIPEMGR.RecipeExport dest,"Proportions",
ESAHMI.ESATAG.WriteValue “Check”,0
```

All the recipes are exported (the third parameter is an empty string) and they are saved in the file indicated in the string variable ‘dest’ which we shall now go on to construct. After the save operation the check bit returns to 0.

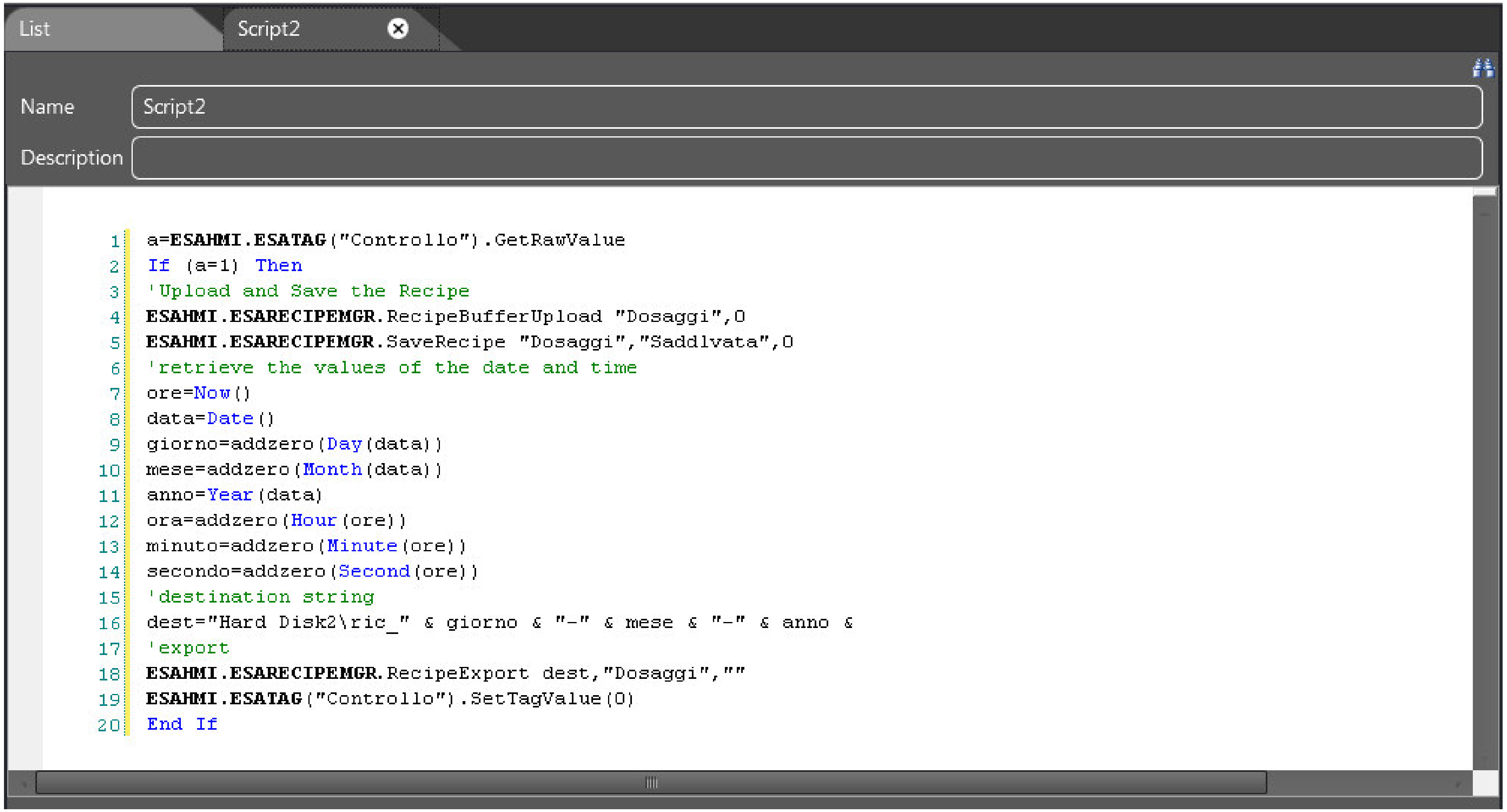
The string ‘dest’ is constructed by adding the details relating to the date and time of the execution of the operation. This information can be obtained using the functions put at the user’s disposal by the programming language, VBScript :

```
time=Now()
date=Date()
day=addzero(Day(date)
month=addzero(Month(date)
year=Year(dat)
hour=addzero(Hour(time)
minute=addzero(Minute(time)
second=addzero(Second(time)
dest="Hard Disk2\ric_" & day & "-" & month & "-" & year &"_h" & hour & "." & minute & "." & second & ".xml"
```

As we can see, the variables day, month, hour, minute and second are passed to the addzero function defined by us in which the zeroes for one-digit values are added.

The final instruction leads to constructing the string ‘dest’ indicating the path and name of the file to which the recipes are exported. In our case, we will save onto the support called ‘Hard Disk2’ (which, for example, could be a USB key) with a name of the type ‘ric_02-12-2005_h12.13.08.xml’. In this way we will be certain to have a series of distinct exportations in a file with unique names in terms of the support.

What follows is an overall view of the Script that has just been configured :



```
a=ESAHMI.ESATAG("Controllo").GetRawValue
If (a=1) Then
'Upload and Save the Recipe
ESAHMI.ESARECIPEMGR.RecipeBufferUpload "Dosaggi",0
ESAHMI.ESARECIPEMGR.SaveRecipe "Dosaggi","Saddlvata",0
'retrieve the values of the date and time
ore=Now()
data=Date()
giorno=addzero(Day(data))
mese=addzero(Month(data))
anno=Year(data)
ora=addzero(Hour(ore))
minuto=addzero(Minute(ore))
secondo=addzero(Second(ore))
'destination string
dest="Hard Disk2\ric_" & giorno & "-" & mese & "-" & anno &
'export
ESAHMI.ESARECIPEMGR.RecipeExport dest,"Dosaggi",""
ESAHMI.ESATAG("Controllo").SetTagValue(0)
End If
```

Example 5 - Canceling all the recipes in the EW

Putting together the methods described in this section you can construct customized functions according to your own project needs. In this example we shall see how to create a function of just a few rows that will cancel all the recipes saved in the EW. This is useful for avoiding cancelling each individual recipe manually and substituting it with a cumulative cancellation.

We also introduce a few rows of code allowing us to 'time' the execution of the entire script (giving us an identifying value of the time taken for it).

Let us now analyze the code:

```
t=Timer()

R_Type="Ten_Var"
```

In the first line we ask for the instant the Script starts (the Timer function returns the number of seconds elapsed since 12:00 AM) and we save this in variable (t).

In the second line we define the recipe type whose instances we want to cancel completely (alternatively we could pass this string value as a parameter for the function, as seen in example 4 for the 'addzero' function).

Next we go and get the name of the first recipe and save it in a variable (a) :

```
a=ESAHMI.ESARECIPEMGR.GetFirstRecipeName(R_Type)
```

if there are no recipes for the type indicated (R_Type), the function returns an empty string (""). Thus cancellation should only occur if the string returned is different from "". We, therefore, use a 'Do While' cycle to make operation :

```
Do While a<>""
ESAHMI.ESARECIPEMGR.DeleteRecipe R_Type,a,0
a=ESAHMI.ESARECIPEMGR.GetFirstRecipeName(R_Type)
Loop
```

As we can see, the ‘While’ cycle remains open until such time as the value of ‘a’ is different from the empty string (that is, until recipes have been saved).

Cancellation occurs in accordance with the type indicated at the beginning of the Script and the current value of a (recipe name); in addition the value 0 is passed to avoid confirmation being asked of the operator.

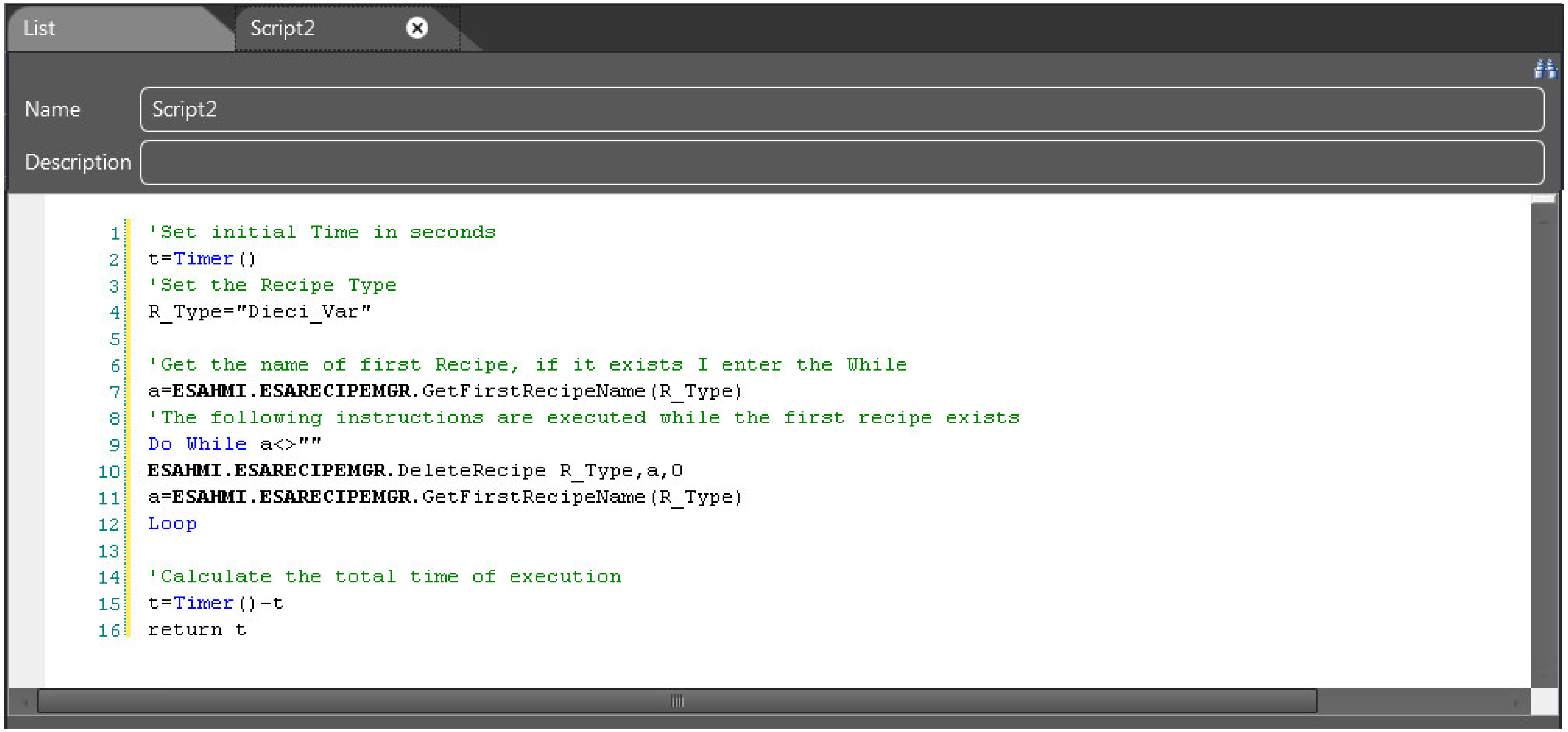
Within the cycle we also update the value of a by getting the new first recipe (we again use GetFirst rather than GetNext because the delete operation has changed the order of the recipes).

Exiting from the While cycle, all the recipes have been eliminated, so all we can do is get the time taken by the Script :

```
t=Timer()-t
return t
```

Using this instruction, the value of t is updated by removing from the current value of Timer() the value obtained at the beginning of the Script (saved in t). Thus, at the end of this instruction t will contain the number of seconds elapsed between the beginning and the end of the cancel operation.

Below is the complete code of our Script:



Name: Script2

```
'Set initial Time in seconds
t=Timer()
'Set the Recipe Type
R_Type="Dieci_Var"

'Get the name of first Recipe, if it exists I enter the While
a=ESAHMI.ESARECIPEMGR.GetFirstRecipeName(R_Type)
'The following instructions are executed while the first recipe exists
Do While a<>""
ESAHMI.ESARECIPEMGR.DeleteRecipe R_Type,a,0
a=ESAHMI.ESARECIPEMGR.GetFirstRecipeName(R_Type)
Loop

'Calculate the total time of execution
t=Timer()-t
return t
```

Example 6: Creates printout of list of recipes

What follows is an example illustrating the use of the print functions. Supposing we want a paper printout of the list of recipes present in the memory of the terminal. The logic behind searching for recipes is similar to that used in the previous example.

Let us first of all initialize the print session using the Start method: by providing parameter 1 the Print options window is shown in runtime before the print session starts.

To abort the print operation the user can click on the X of the window. This control is carried out with an If that checks and, where appropriate, stops the running of all the other code rows :

```
if (ESAHMI.ESAPRN.Start(1)=1) Then
```

Now we create a page heading of a title and two blank rows to separate the title from the contents.

To leave blank rows we use the method WriteLN, passing an empty string. Before writing the title, we set the font at a higher value which we then reduce to a smaller font for the rest of the page.

```
ESAHMI.ESAPRN.FontSize=16
ESAHMI.ESAPRN.WriteLN("Recipe Lists in the EW")
ESAHMI.ESAPRN.WriteLN("")
ESAHMI.ESAPRN.WriteLN("")
ESAHMI.ESAPRN.FontSize=12
```

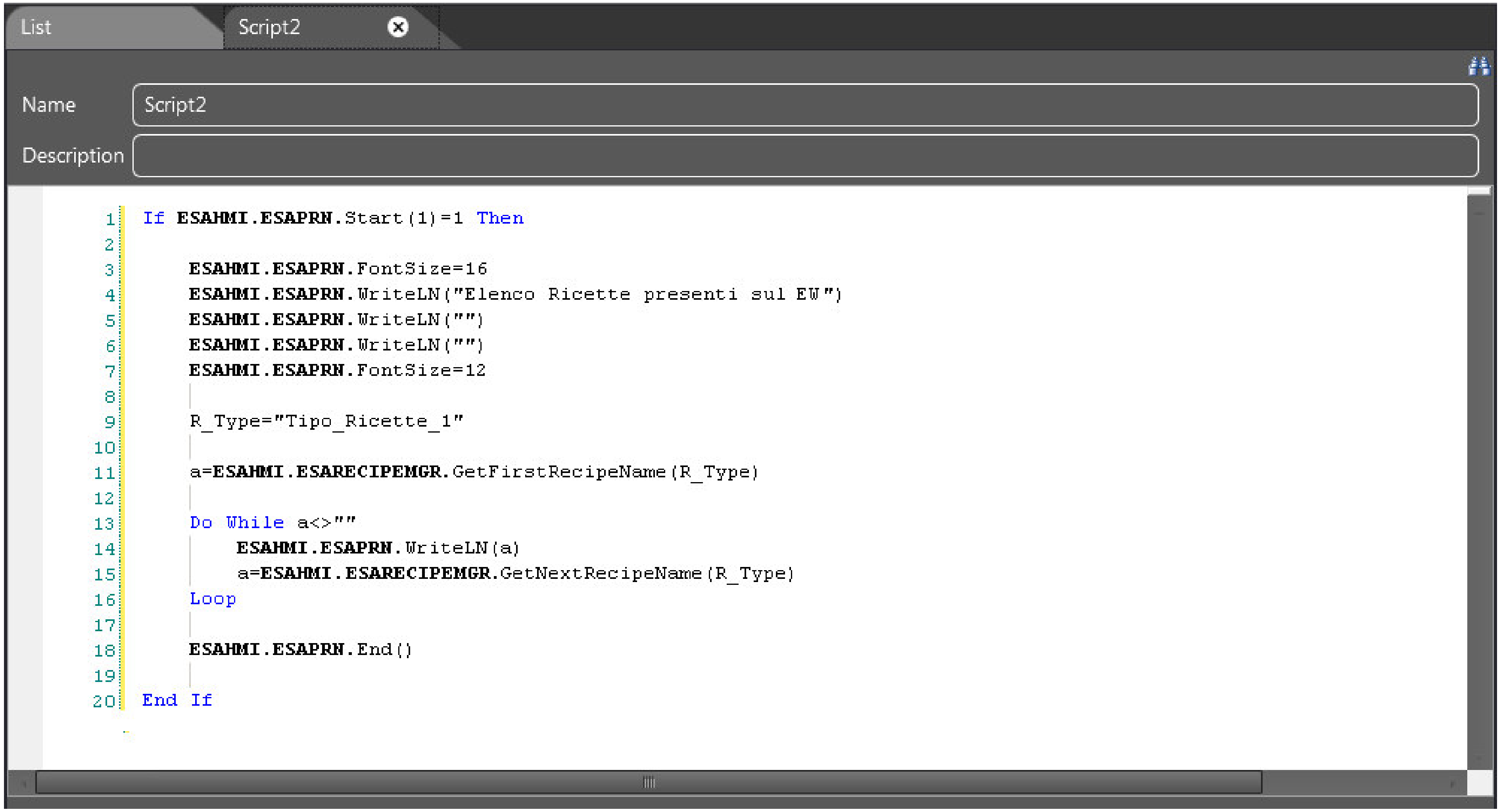
At this point we instance the read-cycle of the recipes saved in the EW using the methods GetFirstRecipeName and GetNextRecipeName. Within the cycle we use the method PrintLN to have the name of a recipe in each line.

```
R_Type="Recipes_Type_1"
a=ESAHMI.ESARECIPEMGR.GetFirstRecipeName(R_Type)Do While a<>""
ESAHMI.ESAPRN.WriteLN(a)
a=ESAHMI.ESARECIPEMGR.GetNextRecipeName(R_Type)
Loop
```

Up to this point we have prepared the contents of the page, now we launch the command that actually starts the printing:

```
ESAHMI.ESAPRN.End()
End If
```

With the execution of this method the print process begins. Below we show the complete text of the Script:



Name: Script2

Description:

```
If ESAHMI.ESAPRN.Start(1)=1 Then

    ESAHMI.ESAPRN.FontSize=16
    ESAHMI.ESAPRN.WriteLN("Elenco Ricette presenti sul EW")
    ESAHMI.ESAPRN.WriteLN("")
    ESAHMI.ESAPRN.WriteLN("")
    ESAHMI.ESAPRN.FontSize=12

    R_Type="Tipo_Ricette_1"

    a=ESAHMI.ESARECIPEMGR.GetFirstRecipeName(R_Type)

    Do While a<>""
        ESAHMI.ESAPRN.WriteLN(a)
        a=ESAHMI.ESARECIPEMGR.GetNextRecipeName(R_Type)
    Loop

    ESAHMI.ESAPRN.End()

End If
```