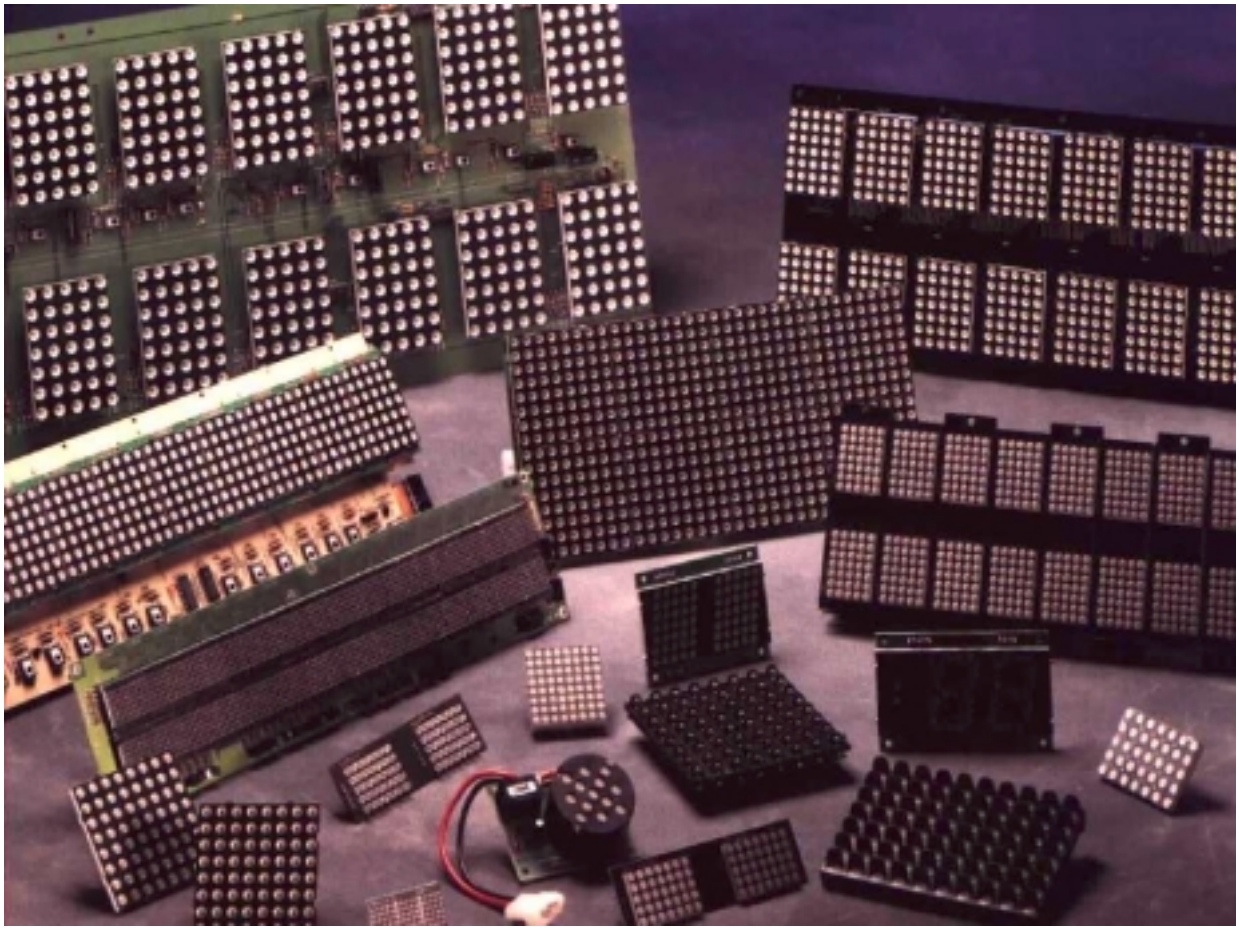
# Software

ALPHA

## High Level Design Document
## ALPHA RT Update Manager

**File Name: User Manual - RT Update Manager.doc**

# RT Update Manager



**Siège européen/European headquarters :**

Adaptive Micro Systems Europe
25, rue Irène Joliot-Curie - F38320 EYBENS
+33 4 76 14 76 00 - Fax : +33 4 76 14 75 70
http://www.ams-e.com ams-e@ams-e.com

**Filiale allemande/German subsidiary :**

Adaptive Micro Systems Deutschland GmbH
Lebacher Str.4 - D66113 Saarbrücken
+49 681 9963 117 - Fax : +49 681 9963 111
licata@ams-e.com

**Siège américain/American headquarters:**

Adaptive Micro Systems Inc.
7840 N. 86th Street - Milwaukee, WI 53224 - USA
1 414 357 2020- Fax : 1 414 357 2029
www.adaptivedisplays.com

# TABLE OF CONTENT



# REVISION HISTORY

| Date | Rev # | Author | Description |
|---|---|---|---|
| 12/11/02 | 1.0 | V.T. | Document creation |
| 03/12/02 | 1.1 | V.T. | Update |
| 23/12/02 | 1.2 | V.T. | SaveConfiguration function added |
| 27/12/02 | 1.3 | V.T. | Update regarding code implementation |
| 13/01/02 | 1.4 | V.T. | Update and function added |
| 22/01/02 | 1.5 | V.T. | Variable function updated |
| 09/04/03 | 1.6 | V.T. | SendFormatedMessageFrame function added |
| 06/06/05 | 1.7 | V.T. | Seed values add for Alpha 2.0 protocol addition |
| | | | |

# 1- Introduction

The purpose of this document is to describe in details the Real Time Update manager.

This application is one element of the ALPHA Display Suite package.
ALPHA Display Suite allows managing the Alpha Adaptive sign range, from the setup to the message building and sending.
For more details of the Suite please refer to: “HDD - ALPHA Display Suite.doc”.

This component has been design as ActiveX to be Excel, Word, Visual Basic and Visual C++ compatible.

This component can be used alone or with XML file generated by others Alpha Display Suite components.

This ActiveX only use XML files, it can not modify them. For this use Sign Manager and Message Manager application (see : “HDD - ALPHA Sign Manager.doc” and “HDD - ALPHA Message Manager.doc”)

## 2- Main Diagram of Alpha Display Suite

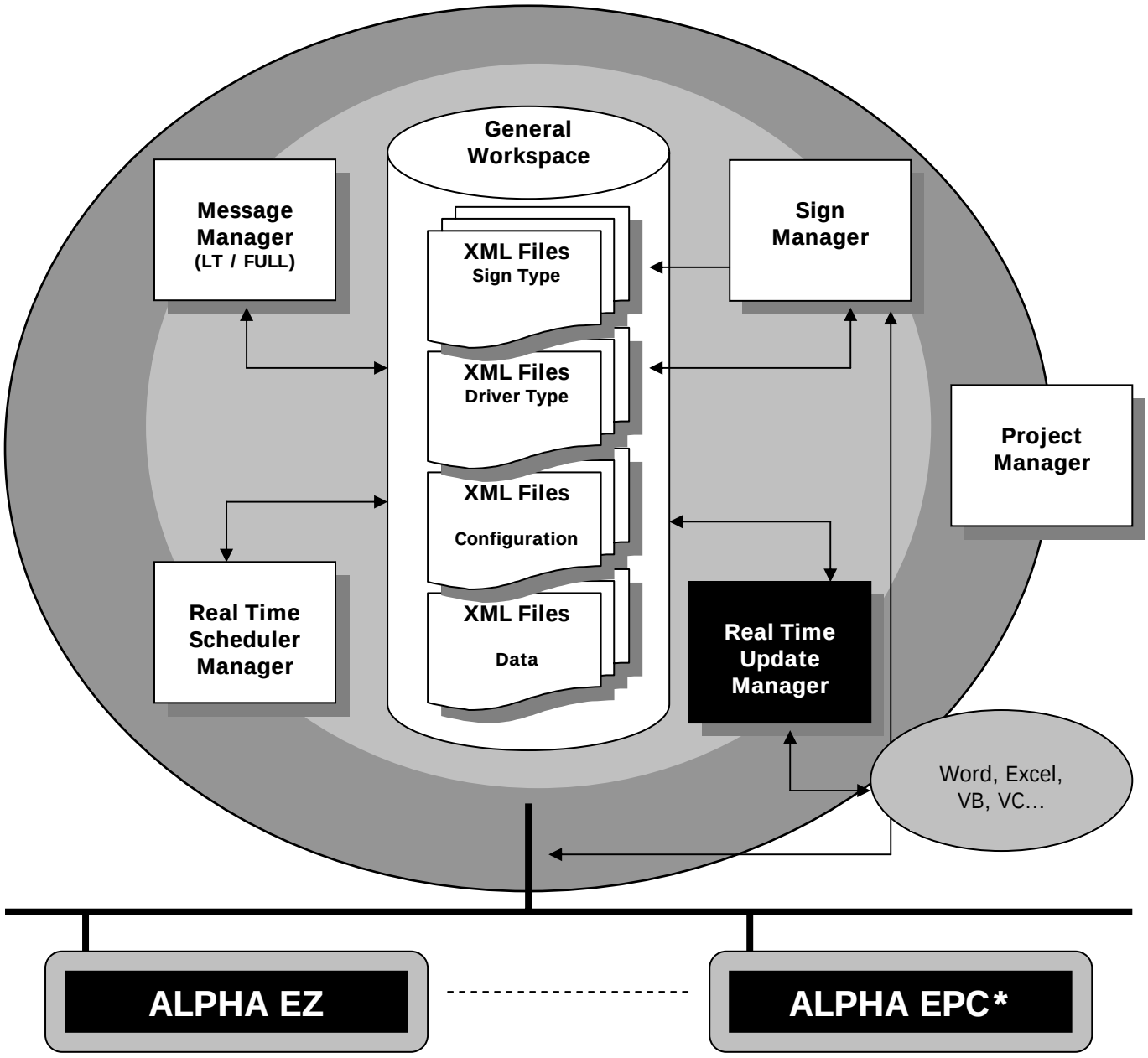



Fig 2.1

*: Not available yet

# 3- Real Time Update Manager ActiveX

## 3.1- Diagram

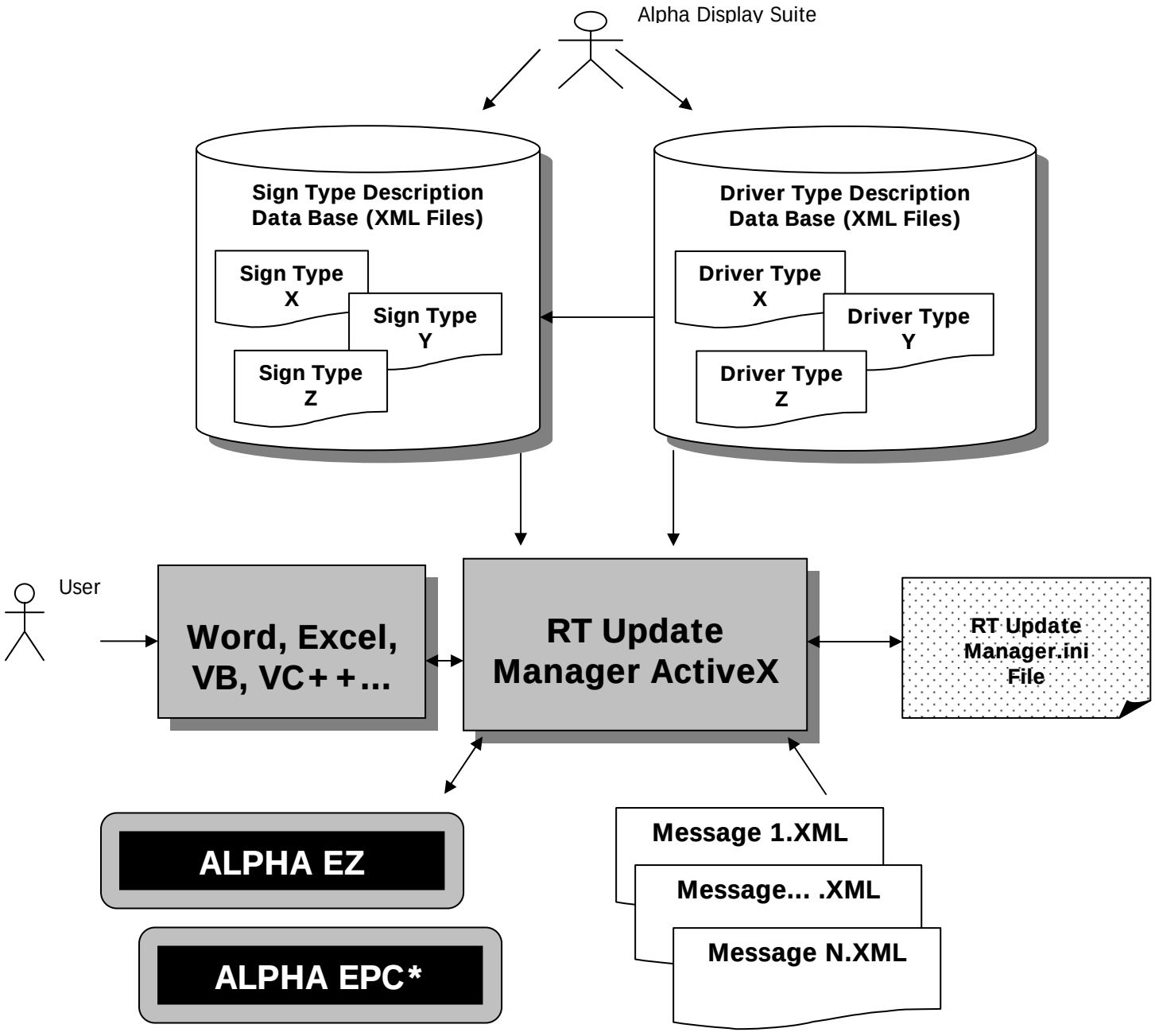



Fig 3.1

*: Not available yet

# Driver functions

## 3.1.1- Serial Driver

### 3.1.1.1- SetSerialComPort

**DOUBLE SetSerialComPort(Short** *comPortNumber***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*comPortNumber* Port number, from 1 to 8

**Remarks:**

Call this function to setup the serial com port number.

### 3.1.1.2- SetSerialBaudRate

**DOUBLE SetSerialComPort(Short** *baudRate***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*baudRate* Available values :

- 4800
- 9600
- 14400
- 19200
- 56000

**Remarks:**

Call this function to setup the serial baud rate.

### 3.1.1.3- SetSerialParity

**DOUBLE SetSerialParity(Short *parity*)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*parity* Available values :

- 0 for None
- 1 for Odd
- 2 for Even

**Remarks:**

Call this function to setup the serial parity.

### 3.1.1.4- SetSerialDataBits

**DOUBLE SetSerialDataBits(Short *dataBits*)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*parity* Available values :

- 7
- 8

**Remarks:**

Call this function to setup the serial data bits.

### 3.1.1.5- SetSerialStopBits

**DOUBLE SetSerialStopBits(Short** *stopBits***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*stopBits* Available values :

- 0
- 2

**Remarks:**

Call this function to setup the serial stop bits.

## 3.1.2- IP Driver

### 3.1.2.1- SetIPAddress

**DOUBLE SetIPAddress(Short** *IPAddress0,* **Short** *IPAddress1,* **Short** *IPAddress2,* **Short** *IPAddress3***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*IPAddress0..3* IP address of your sign.

**Remarks:**

Call this function to specify the IP address of your sign.

### 3.1.2.2- SetIPPort

**DOUBLE SetIPPort(Short** *IPPort***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*IPPort* IP port of your sign.

**Remarks:**

Call this function to specify the IP port of your sign.

## 3.1.3- File Driver

### 3.1.3.1- SetFileName

**DOUBLE SetFileName(LPCTSTR** *fileName***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*filename* File name to store frames.

**Remarks:**

Call this function to specify the file name (and directory) to save the frame send to the sign.

## 3.1.4- Clipboard Driver

### 3.1.4.1- SetClipboardMode

**DOUBLE SetClipBoardMode(Short** *mode***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*mode* Clipboard mode.
Available values:

- 0 for New mode
- 1 for Append mode

**Remarks:**

Call this function to specify the clipboard mode. New mode erases clipboard buffer at each call. Append mode, keep all information in memory.

## 3.1.5- Modem Driver

### 3.1.5.1- SetModemPhoneNumber

**DOUBLE SetModemPhoneNumber(LPCTSTR** *PhoneNumber***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*PhoneNumber* Phone number of your sign.

**Remarks:**

Call this function to specify the phone number of your sign.
Serial functions must been set too. (see: Serial Driver functions)

### 3.1.5.2- SetModemInitString

**DOUBLE SetModemInitString(LPCTSTR** *initString***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*initString* Modem Init string mode.

**Remarks:**

Call this function to specify modem init string; it's depending on your modem model.
Serial functions must been set too. (see: Serial Driver functions)

## 3.1.6- XML file Driver configuration

### 3.1.6.1- SetXMLDriverFileName

**DOUBLE SetXMLDriverFileName(LPCTSTR** *XMLFileName***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*XMLFileName* Xlm file name (with path if needed).

**Remarks:**

Call this function to specify XML file name where the drivers are describe. This file is build via **'Sign Manager' application**. It allows to by pass the manual configuration with 'Serial Driver', 'IP Driver', 'File Driver', 'Clipboard driver' and 'Modem Driver' functions.

## 3.1.7- Sign functions

### 3.1.7.1- SetSignAddress

**DOUBLE SetSignAddress (Short** *signAddress***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*signAddress* Address of the sign. (0 for all signs, ref EZ95 Protocol)

**Remarks:**

Call this function to specify the sign address. (0 by default)

### 3.1.7.2- SetDriverStatus

**DOUBLE SetDriverStatus (LPCTSTR** *driverName,* **Short** *driverStatus***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*driverName* Name of the driver to activate or not.

Available values:

- Serial
- IP
- Modem
- File
- ClipBoard
- XML

*driverStatus* Status of the driver

Available values:

- 0 : Active
- 1 : Inactive

**Remarks:**

Physical drivers (Serial, IP, and modem) are exclusive.
Virtual driver (File and clipboard) can be added.
i.e. : It means that you can validate the driver IP, File, and Clipboard both, but you can not define the IP, and File drivers both.
XML selection is exclusive...

### 3.1.7.3- TestSign

**DOUBLE TestSign()**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

None

**Remarks:**

Call this function to send a message test to know if your activeX configuration is available or not.

# 3.2- Management functions

### 3.2.1.1- ResetSign

**DOUBLE ResetSign()**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

None

**Remarks:**

Call this function to reset the sign define. It's a soft reboot.
Remark, this function does not clear sign memory, for this call ClearMemory function.

### 3.2.1.2- SetTime

**DOUBLE SetTime(Short** hour, **Short** minute**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| hour | Specify the hour to setup (24H mode) |
| minute | Specify the minute to setup |

**Remarks:**

Call this function to setup time, see also SetCurrentTime

### 3.2.1.3- SetCurrentTime

**DOUBLE SetCurrentTime()**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

None

**Remarks:**

Call this function to setup time synchronized with the computer time, see also SetTime

### 3.2.1.4- SetTimeFormat

**DOUBLE SetTimeFormat(LPCTSTR** timeFormat**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

timeFormat

Available values:

- "AMPM"
- "24H"

**Remarks:**

Call this function to setup time format AM/PM or 24H mode

### 3.2.1.5- SetDate

**DOUBLE SetDate(Short** day, **Short** month, **Short** year**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

day Specify the day to setup
Available values:
- 1 : Sunday
- 2 : Monday
- 3 : Tuesday
- Etc...

month Specify the month to setup
- 1 : January
- 2 : February
- 3 : March
- Etc...

year Specify the year to setup i.e : “2002”

**Remarks:**

Call this function to setup date, see also SetCurrentDate.

### 3.2.1.6- SetCurrentDate

**DOUBLE SetCurrentDate()**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

None

**Remarks:**

Call this function to setup time synchronized with the computer date, see also SetDate

### 3.2.1.7- SetDayOfWeek

**DOUBLE SetDayOfWeek(Short** dayOfWeek**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

dayOfWeek Specify the day of week

Available values:

- 1 : Sunday
- 2 : Monday
- 3 : Tuesday
- Etc...

**Remarks:**

Call this function to setup day of week.

### 3.2.1.8- SetSpeakerStatus

**DOUBLE SetSpeakerStatus(Short** speakerStatus**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

speakerStatus Specify the speaker status

Available values:

- 0 : Off
- 1 : On

**Remarks:**

Call this function to set the speaker's status of the sign.

### 3.2.1.9- GenerateSpeakerTone

**DOUBLE GenerateSpeakerTone(Short** beepNumber**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

beepNumber Specify the tone number

Available values:

- 1 : 1 beep
- 3 : 3 Beeps

**Remarks:**

Call this function to generate a tone on the sign.

### 3.2.1.10- SetDimmingPeriod

**DOUBLE SetDimmingPeriod(LPCTSTR** startTime**, LPCTSTR** stopTime**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| startTime | Specify the start time |
| stopTime | Specify the stop time |

Available values: see Time values

**Remarks:**

Call this function to generate a tone on the sign.

### 3.2.1.11- SetTemperatureOffset

**DOUBLE SetTemperatureOffset(Short** *temperatureOffset***, Short** *positionOffset***, Short** s*olarSign***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| *temperatureOffset*<br>*positionOffset* | Structure that specifies the temperature offset parameters<br>Available values:<br>• 0 for negative offset,<br>• 1 for positive offset. |
| *solarSign* | Specifies the sign type. With Solar range you must specify the real temperature, (°F, decimal), with other signs (790i, 460i, 440i and 430i) you must specify a temperature offset.<br>Available values:<br>• 0 for all sign except Solar Sign,<br>• 1 for Solar Sign. |

**Remarks:**

Call this function to set the temperature offset of the sign.

### 3.2.1.12- ClearMemory

**DOUBLE ClearMemory()**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

None

**Remarks:**

Call this function to clear the entire memory of the sign. After this action, the memory is virgin.
See also: ResetSign function

### 3.2.1.13- SetMemoryCfgTableSelfMode

**DOUBLE SetMemoryCfgTableSelfMode()**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

None

**Remarks:**

Call this function to configure the memory automatically for messages and variables with default parameter (size and number).

It is absolutely necessary to call this function or SetMemoryCfgTable function from your application in order to map the sign memory.
See also : SetMemoryCfgTable
The message and variable names are “A”, “B”, “C” etc...
Warning: This function automatically configures priority and background message.

### 3.2.1.14- SetMemorySelfModeParameters

**DOUBLE SetMemorySelfModeParameters**(**Short** messageNumber, **Short** messageSize, **Short** *variableNumber*, **Short** variableSize)

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| messageNumber | Message number (not including priority and background message) |
| messageSize | Message size (octets) |
| variableNumber | Variable number |
| variableSize | Variable size (octets) |

**Remarks:**

Call this function to change the memory parameters.
To save them use the SaveConfiguration. To configure you sign memory call SetMemoryCfgTableSelfMode
SetMemoryCfgTable can be use to.
Warning: This function automatically configures priority and background message.

### 3.2.1.15- SetMemoryCfgTable

**DOUBLE SetMemoryCfgTable**(**Short** messageNumber, **Short** messageSize, **Short** variableNumber, **Short** variableSize)

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

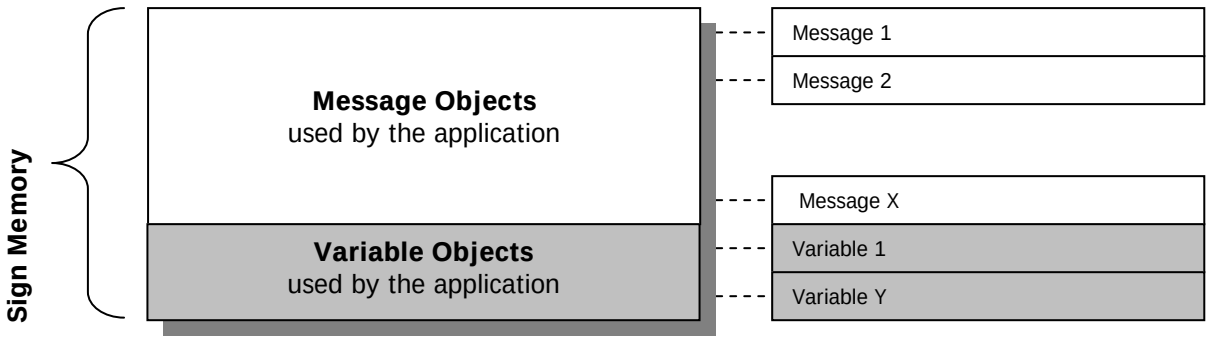
| | |
|---|---|
| messageNumber | Message number |
| messageSize | Message size (octets) |
| variableNumber | Variable number |
| variableSize | Variable size (octets) |

**Remarks:**

Call this function to configure the memory for messages and variables used and handled by the application.

It is absolutely necessary to call this function or SetMemoryCfgTableSelfMode from your application in order to map the sign memory.
See also: SetMemoryCfgTableSelfMode

The message and variable names are “A”, “B”, “C” etc...



### 3.2.1.16- SetRunSequenceTable

**DOUBLE SetRunSequenceTable(LPCTSTR** *runSequence***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*runSequence* Run sequence string

**Remarks:**

Call this function to setup the run sequence i.e : “ABCD”, message A, B, C, D will be displayed
See also: GetRunSequenceTable and TriggerMessage

### 3.2.1.17- GetRunSequenceTable

**LPCTSTR GetRunSequenceTable()**

**Return Value:**

Run sequence string

**Parameters:**

None

**Remarks:**

Call this function to get the run sequence.

### 3.2.1.18- SendFrame

**DOUBLE SendFrame(LPCTSTR** frame**)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

Frame EZ Protocol frame

**Remarks:**

Call this function to send an EZ protocol frame. See EZ Protocol documentation.
This function is used at your own risks...
The frame is encapsulated between <00000><SOH><Sign Type><Sign address> and <EOT>

# 3.3- Message functions

### 3.3.1.1- SendXMLMessage

**DOUBLE SendXMLMessage(LPCTSTR** *XMLFileName,* **LPCTSTR** *messageID***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*XMLFileName* Xlm file name (with path if needed).
*messageID* Message ID ('A' or 'B' or 'C' etc) for this message.

**Remarks:**

Call this function to send a message describe into the XML file. This file is build via **'Message Manager' application**. It allows to by passing all other functions.

### 3.3.1.2- SendXMLMessageGroup

**DOUBLE SendXMLMessageGroup(LPCTSTR** *XMLFileName,* **LPCTSTR** firstM*essageID***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*XMLFileName* Xlm file name (with path if needed).
*messageID* Message ID ('A' or 'B' or 'C' etc) of the first message of the group. Then next message will be the next ID...

**Remarks:**

Call this function to send a message group describe into the XML file. This file is build via **'Message Manager' application**. It allows to by passing all other functions.

### 3.3.1.3- TriggerMessage

**DOUBLE TriggerMessage(LPCTSTR** *messageID*, **SHORT** *messageStatus,* **SHORT** *TriggerMode*)

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*messageID* Message ID (or name) to update status.

*messageStatus* Message status :
Available values:
- 0 : Inactive
- 1 : Active

*TriggerMode* Available values:
- 0 : Momentary. All other message displays are removed.
- 1 : Discrete fix. The messager display are added.

**Remarks:**

Call this function update the status of a message, if status is “Active” the message is display, if “Inactive”, no.
When trigger mode is momentary and the message status is Inactive, the background message is displayed.
See also: SetRunsequenceTable and SetBackgroundMessage function.

### 3.3.1.4- QuickDisplay

**DOUBLE QuickDisplay(LPCTSTR** *message***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*message* Message to display

**Remarks:**

Call this function to display a simple message immediately with default parameters

### 3.3.1.5- QuickDisplayPriorityMessage

**DOUBLE QuickDisplayPriorityMessage(LPCTSTR** *message***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*message* Priority Message to display

**Remarks:**

Call this function to display a simple priority message immediately with default parameters
See also : ResetPriorityMessage

### 3.3.1.6- ResetPriorityMessage

**DOUBLE ResetPriorityMessage()**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*none*

**Remarks:**

Call this function to reset the priority message.
See also : QuickDisplayPriorityMessage

### 3.3.1.7- ClearMessage

**DOUBLE ClearMessage(LPCTSTR** *messagID***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*messageID* Message ID (or name) to clear.

**Remarks:**

Call this function to remove a message.
See also ClearMemory function.

### 3.3.1.8- SendMessage

**DOUBLE SendMessage(LPCTSTR** *message,* **LPCTSTR** *messageID,* **LPCTSTR** *messageColor,* **LPCTSTR** *messageFontStyle,* **LPCTSTR** *messageFontType,* **LPCTSTR** *messageJustification,* **LPCTSTR** *messagePosition,* **LPCTSTR** *messageMode,* **SHORT** *messageSpeed***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| *message* | Message to send. |
| *messageID* | Message ID (or name) to send. |
| *messageColor* | Message color<br>Available values: see Color values |
| *messageFontStyle* | Message font type<br>Available values: see Font type values |
| *messageFontType* | Message font style<br>Available values: see Font Style values |
| *messageJustification* | Message Justification<br>Available values: see Justification values |
| *messagePosition* | Message Position<br>Available values: see Position values |
| *messageMode* | Message Mode<br>Available values: see Mode values |
| *messageSpeed* | Message Speed<br>Available values: see Speed values |

**Remarks:**

Call this function to send a message.
Memory must been setup before, see: SetMemoryCfgTableSelfMode, SetMemoryCfgTable functions.

### 3.3.1.9- DisplayPriorityMessage

**DOUBLE DisplayMessage(LPCTSTR** *priorityMessage,* **LPCTSTR** *priorityMessageColor,* **LPCTSTR** *priorityMessageFontStyle,* **LPCTSTR** *priorityMessageFontType,* **LPCTSTR** *priorityMessageJustification,* **LPCTSTR** *priorityMessagePosition, ,* **LPCTSTR** *priorityMessageMode,* **SHORT** *priorityMessageSpeed***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*priorityMessage* Priority Message to display

see : SendMessage function.

**Remarks:**

Call this function to store a display the priority message.
The message Id of this message is the first one: MSG_ID_0 (“0”)
To remove it use ResetPriorityMessage function.
Memory must been setup before, see: SetMemoryCfgTableSelfMode, SetMemoryCfgTable functions.
These functions automatically configure including MSG_ID_0 message.

### 3.3.1.10- SetBackgroundMessage

**DOUBLE SetBackgroundMessage(LPCTSTR** *backgroundMessage,* **LPCTSTR** *backgroundMessageColor,* **LPCTSTR** *backgroundMessageFontStyle,* **LPCTSTR** *backgroundMessageFontType,* **LPCTSTR** *backgroundMessageJustification,* **LPCTSTR** *backgroundMessagePosition, ,* **LPCTSTR** *backgroundMessageMode,* **SHORT** *backgroundMessageSpeed***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*backgroundMessage* Background Message to display

see : SendMessage function.

**Remarks:**

Call this function to store a background message. This message can be empty.
The message Id of this message is the last one: MSG_ID_80 (“.”)
Use it both with TriggerMessage function.
Memory must been setup before, see: SetMemoryCfgTableSelfMode, SetMemoryCfgTable functions.
These functions automatically configure including MSG_ID_80 message.

### 3.3.1.11- SendFormatedMessageFrame

**DOUBLE SendFormatedMessageFrame (LPCTSTR** *formatedMessageFrame***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

*formatedMessageFrame :* Message to display formatted regarding the protocol defined below.

see : SendMessage function.

**Remarks:**

This function could be used to send message from Microsoft Word or Excel application.

**Protocol :**

The frame send is a concatenation of specific parameter. We can distinguish five kind of parameters:

*Message parameters:* to specify the message properties, as its name, its periodicity.
*Page parameters:* to specify the page properties, as its position, its mode...
*Region parameters:* to specify the region properties, as its justification...
*Text parameters:* to specify the text properties, as its fonts type, its size...
*Object parameters:* to add specific object , as date, time...

**Message parameters**

**[M]***Message Name;StartTime;StopTime;StartDay;StopDay***[M]**

| | |
|---|---|
| *Message Name* | Available values: see Message and variable ID values |
| *StartTime* | Available values: see Time values (optional regarding Periodicity value selected) |
| *StopTime* | Available values: see Time values (optional regarding Periodicity value selected) |
| *StartDay* | Available values: see Date values (optional regarding Periodicity value selected) |
| *StopDay* | Available values: see Date values (optional regarding Periodicity value selected) |

**Page parameters**

**[P]***Position;Mode;Duration***[P]**

| | |
|---|---|
| *Position* | Available values: see Position values |
| *Mode* | Available values: see Mode values |
| *Duration* | Available values: see Speed values |

**[P]***New Page***[P]**

| | |
|---|---|
| *New Page* | Available value: NEW_PAGE |

**Region parameters**

**[R]***Justification***[R]**

*Justification* Available values: see Justification values

**[R]***New Line***[R]**

*New Line* Available value: NEW_LINE

**Text parameters**

**[T]***Font Style***[T]**

*Font Type* Available values: Font Style values

**[T]***Font Type***[T]**

*Font Type* Available values: Font Type values

**[T]***Color***[T]**

Available values: Color values

**Object parameters**

**[O]***Time***[O]**

*Time* Available value: TIME

**[O]***Date***[O]**

*Date* Available value: Date Type values

**[O]***Temperature***[O]**

*Temperature* Available value: Temperature values
See SetTimeFormat function to switch between 24h-AM/PM format.

**[O]***Variable name***[O]**

*Variable Name* Available values: see Message and variable ID values

Extended frame example:

# 3.4- Variable functions

### 3.4.1.1- UpdateTextVariable

**DOUBLE UpdateTextVariable(LPCTSTR** *variableID,* **LPCTSTR** *variableText,* **LPCTSTR** *variableColor,* **LPCTSTR** *variableStyle***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| *variableID* | Variable ID (or name) to update |
| *variableText* | New Variable text. |
| *variableColor* | Variable color<br>Available values: see Color values |
| *variableFontStyle* | Variable font style<br>Available values: see Font style values |

**Remarks:**

Call this function to update a text variable.
Be careful, memory must be configured before. See: SetMemoryCfgTableSelfMode, SetMemoryCfgTable function.
Variable must be called from a message. This message must be formatted from other applications (ex message manager) or directly by sending the corresponding frame via SendFrame or SendMessage (see Alpha Protocol).

### 3.4.1.2- UpdateIntegerVariable

**DOUBLE UpdateIntegerVariable(LPCTSTR** *variableID,* **Double** *variableValue,* **LPCTSTR** *variablePadding, ,* **Double** *variableDisplaysize,* **LPCTSTR** *variableColor,* **LPCTSTR** *variableStyle***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| *variableID* | Variable ID (or name) to update |
| *variableValue* | New Variable value |
| *variablePadding* | Padding property, authorized values:<br>- "Trailing Spaces"<br>- "Leading Spaces"<br>- "Leading Zeros"<br>- "None" |
| *variableDisplaysize* | Variable display size. See example bellow. |
| *variableColor* | Variable color<br>Available values: see Color values |
| *variableFontStyle* | Variable font style<br>Available values: see Font style values |

Example: with a *variableDisplaysize = 8.*

*If variablePadding =* "Trailing Spaces", '12345' variable becomes: "12345___"
*If variablePadding =* "Leading Spaces", '12345' variable becomes: "___12345
*If variablePadding =* "Leading Zeros", '12345' variable becomes: "00012345"
*If variablePadding =* "None", '12345' variable stays: "12345" (*variableDisplaysize parameter* is inactive*)*

**Remarks:**

Call this function to update a text variable.
Be careful, memory must be configured before. See: SetMemoryCfgTableSelfMode, SetMemoryCfgTable function.
Variable must be called from a message. This message must be formatted from other applications (ex message manager) or directly by sending the corresponding frame via SendFrame or SendMessage (see Alpha Protocol).

### 3.4.1.3- UpdateInteger16Variable

**DOUBLE UpdateIntegerVariable(LPCTSTR** *variableID,* **Double** *variableValue,* **LPCTSTR** *variablePadding, ,* **Double** *variableDisplaysize,* **LPCTSTR** *variableColor,* **LPCTSTR** *variableStyle***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

Same function and parameters than UpdateIntegerVariable function, but the *variableValue* is displayed on hexadecimal format.

### 3.4.1.4- UpdateFloatVariable

**DOUBLE UpdateFloatVariable(LPCTSTR** *variableID,* **Double** *variableValue,* **LPCTSTR** *variablePadding,* **Double** *variableDisplaysize,* **Double** *variablePrecision,* **LPCTSTR** *variableColor,* **LPCTSTR** *variableStyle***)**

**Return Value:**

- **SUCCESS**
- Standard EZProtocol error, see also GetErrorDescription Function

**Parameters:**

| | |
|---|---|
| *variableID* | Variable ID (or name) to update |
| *variableValue* | New Variable value |
| *variablePadding* | Padding property, authorized values:<br>- "Trailing Spaces"<br>- "Leading Spaces"<br>- "Leading Zeros"<br>- "None" |
| *variableDisplaysize* | Variable display size. See example bellow. |
| *variablePrecision* | Variable precision after. See example bellow. |
| *variableColor* | Variable color<br>Available values: see Color values |
| *variableFontStyle* | Variable font style<br>Available values: see Font style values |

Example: with a *variableDisplaysize = 10. and variablePrecision=1*

*If variablePadding =* "Trailing Spaces", '12345,12345' variable becomes: "12345,1___"
*If variablePadding =* "Leading Spaces", '12345,12345' variable becomes: "___12345,1
*If variablePadding =* "Leading Zeros", '12345,12345'' variable becomes: "00012345,1"
*If variablePadding =* "None", '12345,12345' variable becomes: "12345,1" (*variableDisplaysize parameter* is inactive*)*

**Remarks:**

Call this function to update a text variable.
Be careful, memory must be configured before. See: SetMemoryCfgTableSelfMode, SetMemoryCfgTable function.
Variable must be called from a message. This message must be formatted from other applications (ex message manager) or directly by sending the corresponding frame via SendFrame or SendMessage (see Alpha Protocol).

# 3.5- Management functions

### 3.5.1.1- SaveConfiguration

**Double SaveConfiguration()**

**Return Value:**

- **SUCCESS**

**Parameters:**

*none*

**Remarks:**

Call this function to save driver parameters.
Be careful only 8N1 or 7E2 parameters are saved for Serial driver (8 bit impose 8N1 and 7 impose 7E2)

### 3.5.1.2- GetErrorDescription

**LPCTSTR GetErrorDescription(double** *errorValue***)**

**Return Value:**

Description string of the error

**Parameters:**

*errorValue* Error value return by functions

**Remarks:**

Call this function to get a translation of current t error return by most of RT ActiveX functions. Several errors are specific to this ActiveX others come from EZ Protocol SDK. (See relative documentation)

### 3.5.1.3- SetAcknowledgement

**Double SetAcknowledgement(Short** *acknowledgement***)**

**Return Value:**

- **SUCCESS**

**Parameters:**

*acknowledgement* Acknowledgement status :
Available values:
- 0 : Inactive
- 1 : Active

**Remarks:**

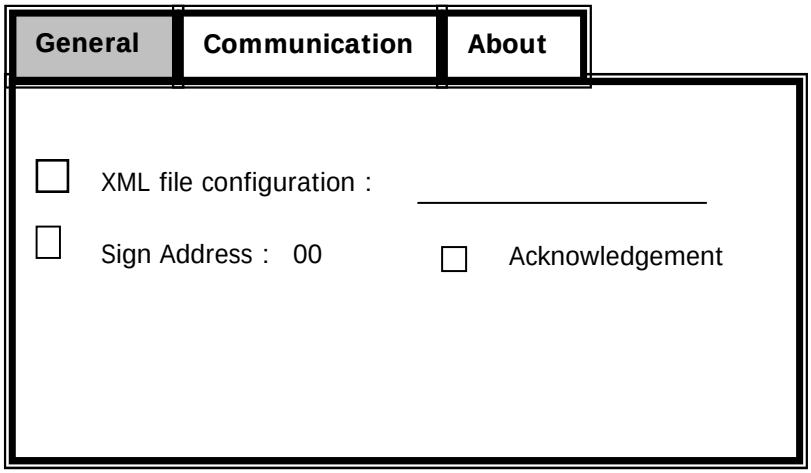
Call this function to validate or invalidate the acknowledgement function.
Warning, if the acknowledgement is activated, the response time is about two times more that without...

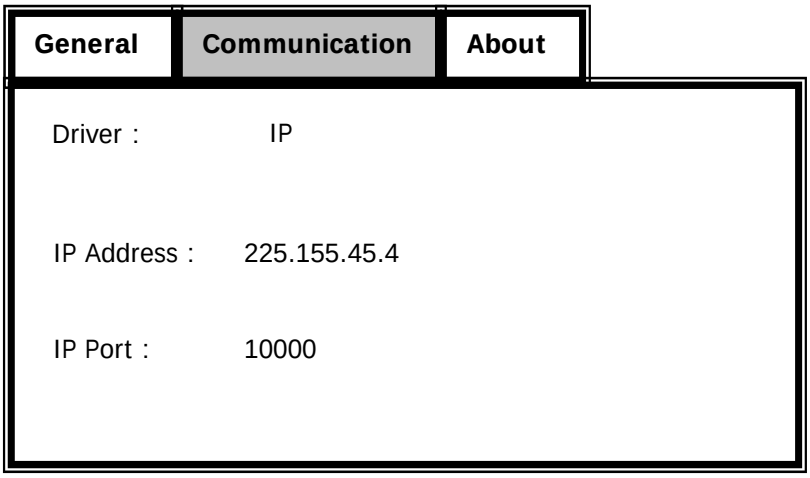
## 3.6- Property screen

This property screen allows setup directly the active without using Drivers and Sign functions.
This screen is accessible via a Visual Basic project.

### 3.6.1- General property screen

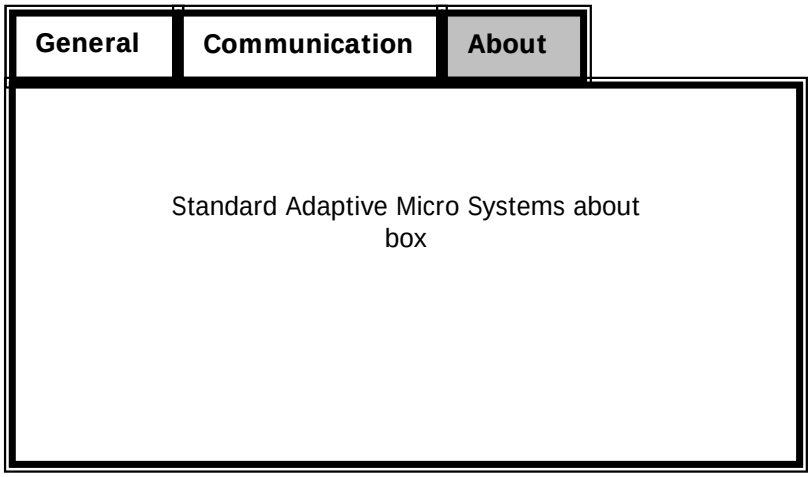


### 3.6.2- Communication property screen




Idem for other drivers...

### 3.6.3- About property screen



# 4- Index

## 4.1- Message and variable ID values

A • B • C • D • E • F • G • H • I • J • K • L • M • O • P • Q • R • S • T • U • V • W • X • Y • Z •
• a • b • c • d • e • f • g • h • i • j • k • l • m • n • o • p • q • r • s • t • u • v • w • x • y • z •
• 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9 • + • - • * • / • = • < • > • ( • ) • [ • ] • ! • # • & • %• $ • ; • : • . •

## 4.2- Time values

• TIME_12_00_AM • TIME_12_10_AM • TIME_12_20_AM • TIME_12_30_AM • TIME_12_40_AM •
• TIME_12_50_AM • TIME_01_00_AM • TIME_01_10_AM • TIME_01_20_AM • TIME_01_30_AM •
• TIME_01_40_AM • TIME_01_50_AM • TIME_02_00_AM • TIME_02_10_AM • TIME_02_20_AM •
• TIME_02_30_AM • TIME_02_40_AM • TIME_02_50_AM • TIME_03_00_AM • TIME_03_10_AM •
• TIME_03_20_AM • TIME_03_30_AM • TIME_03_40_AM • TIME_03_50_AM • TIME_04_00_AM •
• TIME_04_10_AM • TIME_04_20_AM • TIME_04_30_AM • TIME_04_40_AM • TIME_04_50_AM •
• TIME_05_00_AM • TIME_05_10_AM • TIME_05_20_AM • TIME_05_30_AM • TIME_05_40_AM •
• TIME_05_50_AM • TIME_06_00_AM • TIME_06_10_AM • TIME_06_20_AM • TIME_06_30_AM •
• TIME_06_40_AM • TIME_06_50_AM • TIME_07_00_AM • TIME_07_10_AM • TIME_07_20_AM •
• TIME_07_30_AM • TIME_07_40_AM • TIME_07_50_AM • TIME_08_00_AM • TIME_08_10_AM •
• TIME_08_20_AM • TIME_08_30_AM • TIME_08_40_AM • TIME_08_50_AM • TIME_09_00_AM •
• TIME_09_10_AM • TIME_09_20_AM • TIME_09_30_AM • TIME_09_40_AM • TIME_09_50_AM •
• TIME_10_00_AM • TIME_10_10_AM • TIME_10_20_AM • TIME_10_30_AM • TIME_10_40_AM •
• TIME_10_50_AM • TIME_11_00_AM • TIME_11_10_AM • TIME_11_20_AM • TIME_11_30_AM •
• TIME_11_40_AM • TIME_11_50_AM • TIME_12_00_PM • TIME_12_10_PM • TIME_12_20_PM •
• TIME_12_30_PM • TIME_12_40_PM • TIME_12_50_PM • TIME_01_00_PM • TIME_01_10_PM •
• TIME_01_20_PM • TIME_01_30_PM • TIME_01_40_PM • TIME_01_50_PM • TIME_02_00_PM •
• TIME_02_10_PM • TIME_02_20_PM • TIME_02_30_PM • TIME_02_40_PM • TIME_02_50_PM •
• TIME_03_00_PM • TIME_03_10_PM • TIME_03_20_PM • TIME_03_30_PM • TIME_03_40_PM •
• TIME_03_50_PM • TIME_04_00_PM • TIME_04_10_PM • TIME_04_20_PM • TIME_04_30_PM •
• TIME_04_40_PM • TIME_04_50_PM • TIME_05_00_PM • TIME_05_10_PM • TIME_05_20_PM •
• TIME_05_30_PM • TIME_05_40_PM • TIME_05_50_PM • TIME_06_00_PM • TIME_06_10_PM •
• TIME_06_20_PM • TIME_06_30_PM • TIME_06_40_PM • TIME_06_50_PM • TIME_07_00_PM •
• TIME_07_10_PM • TIME_07_20_PM • TIME_07_30_PM • TIME_07_40_PM • TIME_07_50_PM •
• TIME_08_00_PM • TIME_08_10_PM • TIME_08_20_PM • TIME_08_30_PM • TIME_08_40_PM •
• TIME_08_50_PM • TIME_09_00_PM • TIME_09_10_PM • TIME_09_20_PM • TIME_09_30_PM •
• TIME_09_40_PM • TIME_09_50_PM • TIME_10_00_PM • TIME_10_10_PM • TIME_10_20_PM •
• TIME_10_30_PM • TIME_10_40_PM • TIME_10_50_PM • TIME_11_00_PM • TIME_11_10_PM •
• TIME_11_20_PM • TIME_11_30_PM • TIME_11_40_PM • TIME_11_50_PM • TIME_ALL_DAY •
• TIME_NEVER • ALWAYS •

If TIME_ALL_DAY, TIME_NEVER or TIME_ALWAYS are selected, second value is not taking in account.

## 4.3- Date values

• DATE_DAILY • DATE_SUNDAY • DATE_MONDAY • DATE_TUESDAY • DATE_WEDNESDAY •
• DATE_THURSDAY • DATE_FRIDAY • DATE_SATURDAY • DATE_MONDAY_FRIDAY •
• DATE_WEEKENDS• DATE_ALWAYS • DATE_NEVER •

## 4.4- Color values

• COLOR_DEFAULT • COLOR_RED • COLOR_GREEN • COLOR_AMBER • COLOR_DIM_RED •
• COLOR_DIM_GREEN • COLOR_BROWN • COLOR_ORANGE • COLOR_YELLOW •
• COLOR_RAINBOW1 • COLOR_RAINBOW2 • COLOR_COLORMIX • COLOR_AUTOCOLOR •

## 4.5- Font Style values

• STYLE_ DEFAULT • STYLE_DOUBLE_HIGH_ON • STYLE_DOUBLE_HIGH_OFF •
• STYLE_TRUE_DESCENDERS_ON • STYLE_TRUE_DESCENDERS_OFF • STYLE_FLASH_ON •
• STYLE_FLASH_OFF • STYLE_WIDE_ON • STYLE_WIDE_OFF • STYLE_DOUBLE_WIDE_ON •
• STYLE_DOUBLE_WIDE_OFF •

## 4.6- Font Type values

• TYPE _DEFAULT • TYPE_STD_5 • TYPE_STD_7 • TYPE_STD_10 • TYPE_STD_FULL •
• TYPE_FANCY_7 • TYPE_FANCY_FULL •

## 4.7- Justification values

• JUSTIFICATION_DEFAULT • JUSTIFICATION_CENTER • JUSTIFICATION_LEFT •

## 4.8- Position values

• POSITION_DEFAULT • POSITION_BOTTOM • POSITION_FILL • POSITION_MIDDLE •
• POSITION_TOP •

## 4.9- Mode values

• MODE_DEFAULT • MODE_AUTOMODE • MODE_FLASH • MODE_HOLD • MODE_ROLL_UP •
• MODE_ROLL_DOWN • MODE_ROLL_LEFT • MODE_ROLL_RIGHT • MODE_ROTATE_STANDARD •
• MODE_ROTATE_CONDENSED • MODE_SCROLL • MODE_INTERLOCK • MODE_SLIDE •
• MODE_SNOW • MODE_SPARKLE • MODE_SPRAY • MODE_STARBURST • MODE_SWITCH •
• MODE_TWINKLE • MODE_WIPE_UP • MODE_WIPE_DOWN • MODE_WIPE_LEFT •
• MODE_WIPE_RIGHT • MODE_WIPE_IN • MODE_WIPE_OUT •

## 4.10- Date Type values

• DATE_SLASH_MMDDYY • DATE_SLASH_DDMMYY • DATE_DASH_MMDDYY •
• DATE_DASH_DDMMYY • DATE_DOT_MMDDYY • DATE_DOT_DDMMYY • DATE_SPACE_MMDDYY •
• DATE_SPACE_DDMMYY • DATE_MMMDDYYYY • DATE_DAY_OF_WEEK •

## 4.11- Temperature values

• TEMPERATURE_CELSIUS • TEMPERATURE_FAHRENHEIT •

## 4.12- Speed values

• -1 (default) • 0 • 1 • 2 • 3 • 4 • 5 •

Remark: With 4000 Premier, 9000 Premier and Eclipse it's possible to use directly the time value between 000 second to 255 seconds. (Always use 3 characters)

# NOTES