



DeepL

Subscribe to DeepL Pro to translate larger documents.
Visit www.DeepL.com/pro for more information.

AIRHMI LCD SCREEN EDITOR MANUAL

AirHMI LCD DISPLAY EDITOR GUIDE

AIRHMI

Version: 2.2

AIRHMI LCD SCREEN EDITOR MANUAL

AirHMI Visual Screen Creator is designed to create Human Machine Interface GUIs for AirHMI LCD displays in the most efficient time with the highest level of design satisfaction. In the editor we have functionalities from the world of Design and Programming: In addition to the screen design support that you can be original from the visually rich object treasure and you can easily create according to your wishes, it also provides many convenience to the user in the programming part.

AIRHMI

AIRHMI LCD SCREEN EDITOR MANUAL

History	Function Name	Firmware Version
05.05.2024	Convert_IntToString	4.00
05.05.2024	Convert_FloatToString	4.00
05.05.2024	Convert_StringToInt	4.00
05.05.2024	Convert_StringToFloat	4.00
28.07.2014	Transhape specs added.	4.04
28.07.2014	Toggle specs added.	4.04
07.10.2014	StructSet and StructGet added.	4.05

AIRHMI

AIRHMI LCD SCREEN EDITOR MANUAL

TABLE OF CONTENTS

1.	AirHMI Visual Screen Creator INSTALLATION	1
2.	PROJECT CREATION.....	2
3.	DEVICE CONNECTION.....	4
4.	AirHMI EDITOR MAIN INTERFACE	5
4.1	TITLE BAR.....	5
4.2	MAIN MENU and TOOL BARS.....	5
4.3	COMPONENTS COMPARTMENT	8
4.4	SCREEN / COMMAND TAB	9
4.5	DESIGN MAIN SCREEN AREA	10
4.6	AREA OF COMPONENTS WITHOUT VISUALS.....	11
4.7	ATTRIBUTE SPACE OF OBJECTS.....	11
4.8	3.7.1 Display Area of Objects Used in the Project.....	11
4.9	3.7.2 Attributes of Objects Display / Setting Area.....	12
4.10	ATTRIBUTES DESCRIPTION FIELD	12
4.11	USER PROJECT CODE MENU and TOOL BARS	12
4.12	USER PROJECT CODE FIELD.....	12

AIRHMI LCD SCREEN EDITOR MANUAL

4.13	CODE AREA ZOOM AREA.....	13
4.14	CODE FIELD.....	13
5.	Transformations of Variables into Each Other.....	15
5.1	Convert an Integer Expression to String (Char Array).....	15
5.2	Using sprintf.....	17
5.3	atoi.....	18
5.4	atof.....	19
6.	ARHMIC OBJECTS AND FUNCTIONS.....	20
6.1	TIMER	20
6.2	Button	23
6.3	Label.....	29
6.4	Image.....	34
6.5	ProgressBar.....	39
6.6	Slider	44
6.7	Gauge	49
6.8	ListView	54
6.9	ListWheel	63

AIRHMI LCD SCREEN EDITOR MANUAL

6.10	TransShape	68
6.11	Toggle.....	70
6.12	Graph.....	76
6.13	Variable	81
6.14	Delay().....	91
6.15	uartDataGet().....	92
6.16	ChangeScreenSet ().....	93
6.17	dateSet ()	94
6.18	timeSet ().....	95
6.19	dateGet ().....	96
6.20	timeGet ()	97
6.21	AudioPlay().....	98
6.22	AudioStop().....	99
6.23	AudioStatusGet().....	100
6.24	VideoPlay()	101
6.25	Video_Play_XY().....	102
6.26	File_write ().....	104

AIRHMI LCD SCREEN EDITOR MANUAL

6.27	File_read().....	105
6.28	File_size().....	106
6.29	GPIO_Write().....	107
6.30	GPIO_Read().....	108
6.31	PWM_Set().....	109
6.32	BuzzerSet().....	110
6.33	I2C_Write().....	111
6.34	I2C_Read ().....	112
6.35	millis().....	113
6.36	KeypadAlpha().....	114
6.37	Modbus_ReadHoldingRegisters().....	115
6.38	Modbus_WriteSingleRegister().....	117
6.39	Modbus_WriteMultipleRegisters().....	119
6.40	Modbus_ReadInputRegisters().....	121
7.	Ethernet	123
7.1	Dhcp & Static ip identification.....	123
7.2	IP Address Inquiry.....	125

AIRHMI LCD SCREEN EDITOR MANUAL

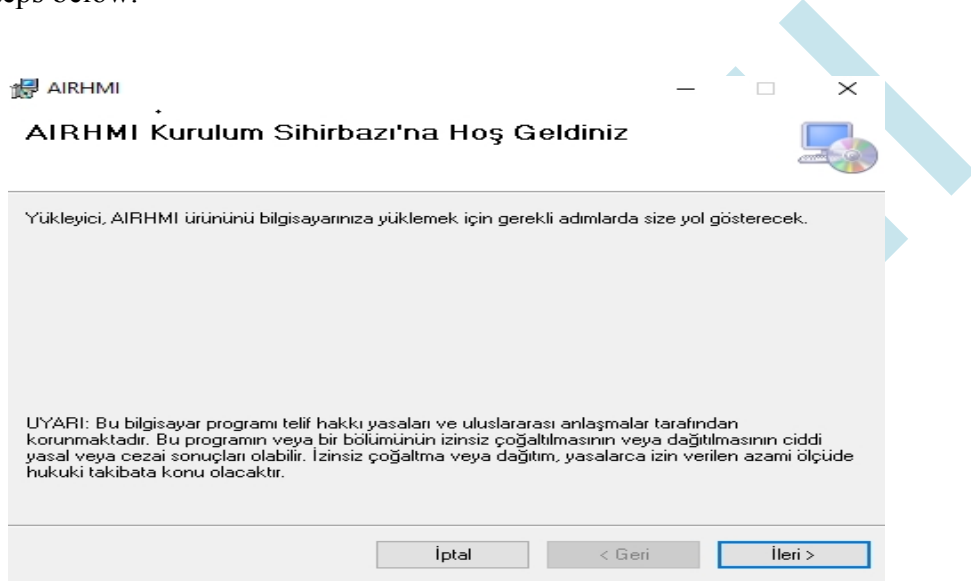
7.3	MAC Address Inquiry	126
7.4	Ethernet TCP Socket Connection	127
7.5	Ethernet TCP Socket Send Receive	128
7.6	Send Ethernet TCP Socket	129
7.7	Ethernet TCP Socket Al	130
7.8	Ethernet TCP Socket Close	131
7.9	Ethernet TCP Socket Status Query	132
7.10	http post and get	133
8.	Libraries	135
8.1	stdio.h	135
8.2	stdlib.h	136
8.3	math.h	138
8.4	string.h	141

AIRHMI LCD SCREEN EDITOR MANUAL

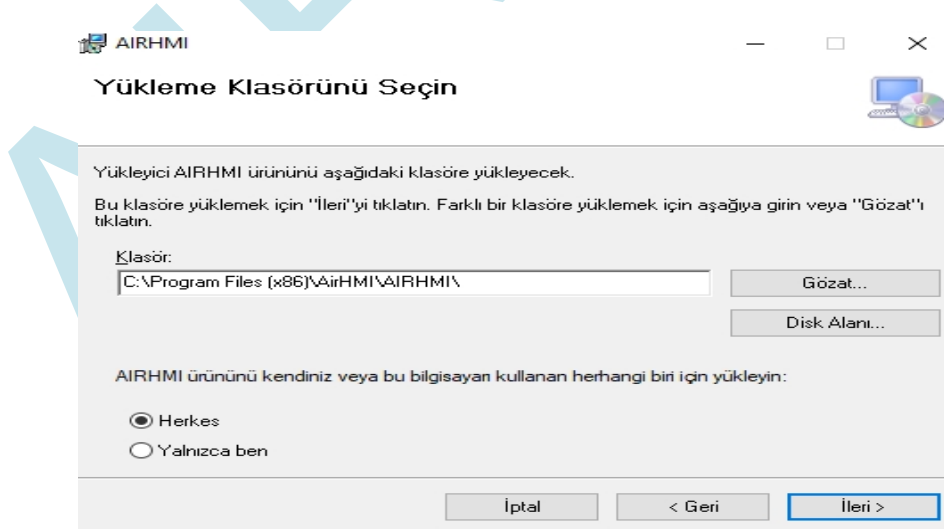
1. AirHMI Visual Screen Creator INSTALLATION

Download Link: <https://www.airhmi.com/airhmi-visualcreator>

Double click on AIRHMISSETUP.msi to install AirHMI Editor on your computer. After that, follow the steps below.



Select the installation folder and other options as desired and press next to start the installation.

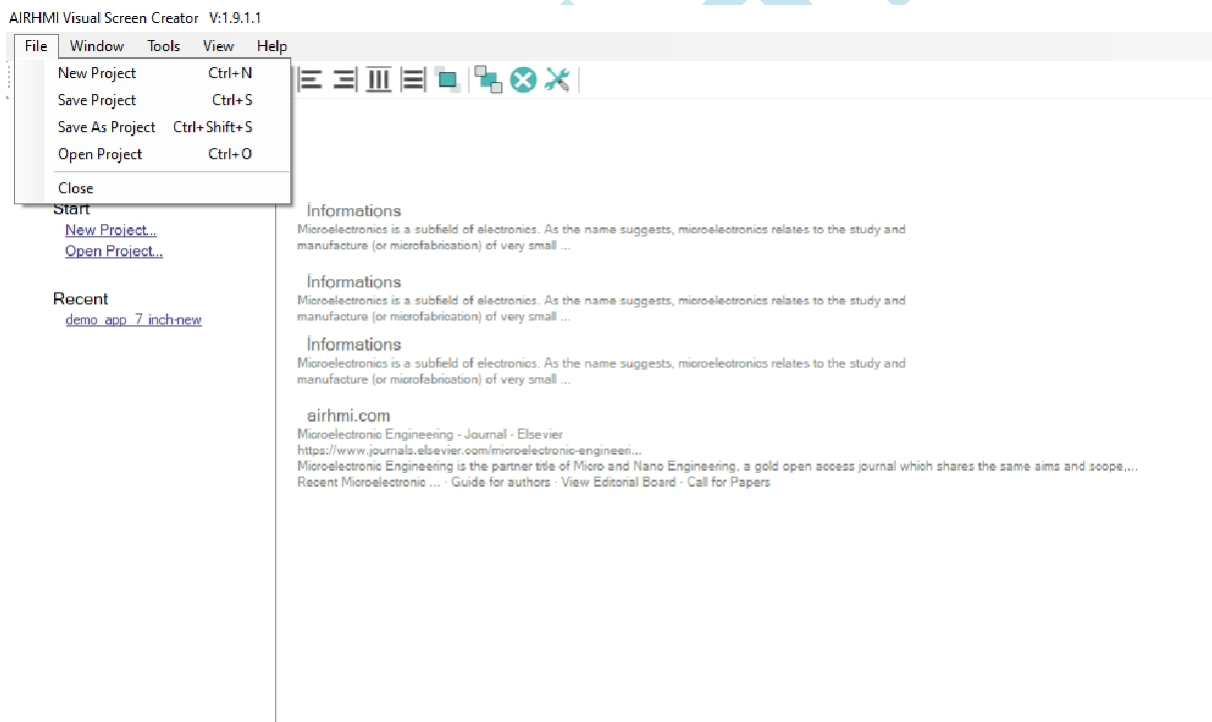


AIRHMI LCD SCREEN EDITOR MANUAL

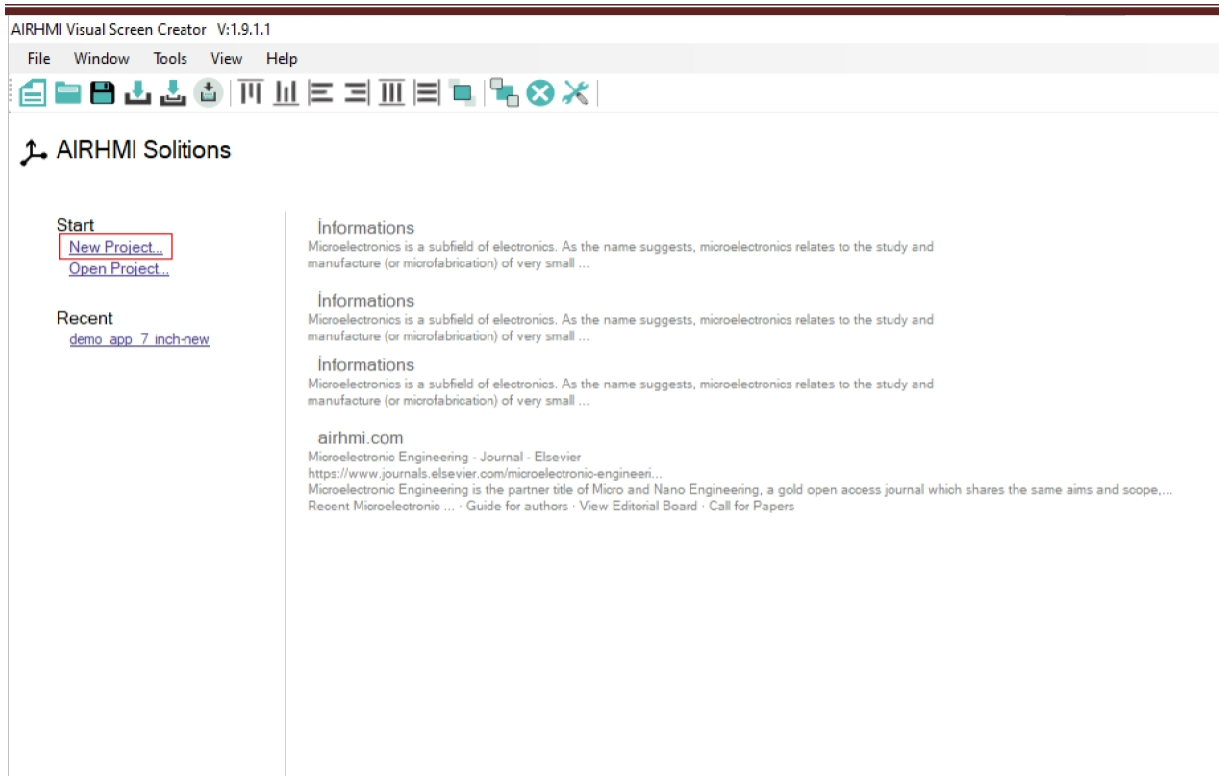
2. PROJECT CREATION

To create an interface with AirHMI, you must first download and install the AirHMI Editor program on your computer. The drag-and-drop feature in the AirHMI Editor program makes interface development easier. With AirHMI Editor, you can add buttons, pictures, text, Progress bar, Gauge, Key, Numeric inputs and outputs to see Analog and Digital values. You can add many components such as.

The program is very easy to install. After installation, you should run the AirHMI Editor program. You will see a page as shown in the pictures below. From this page, follow the File - New path in the upper left corner or the first opening of the program. You create your project by clicking on New Project from the tabs that appear on the page.



AIRHMI LCD SCREEN EDITOR



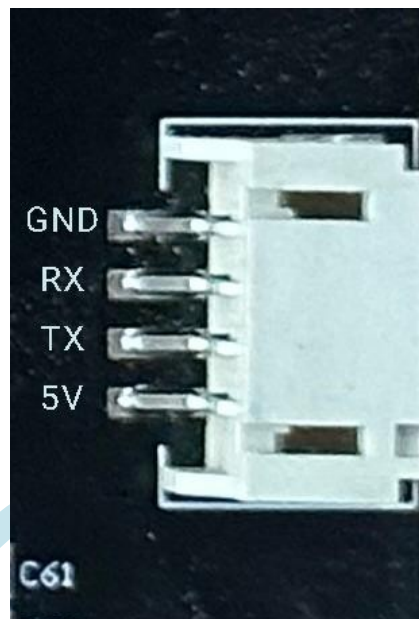
After registration, you will see a page as shown in the picture below. On the page that appears, settings related to the size and resolution of the screen should be made.



3. DEVICE CONNECTION

The power connector that we energize the AirHMI display has four pins. 1 and 4 are supply, the middle two pins are uart communication pins.

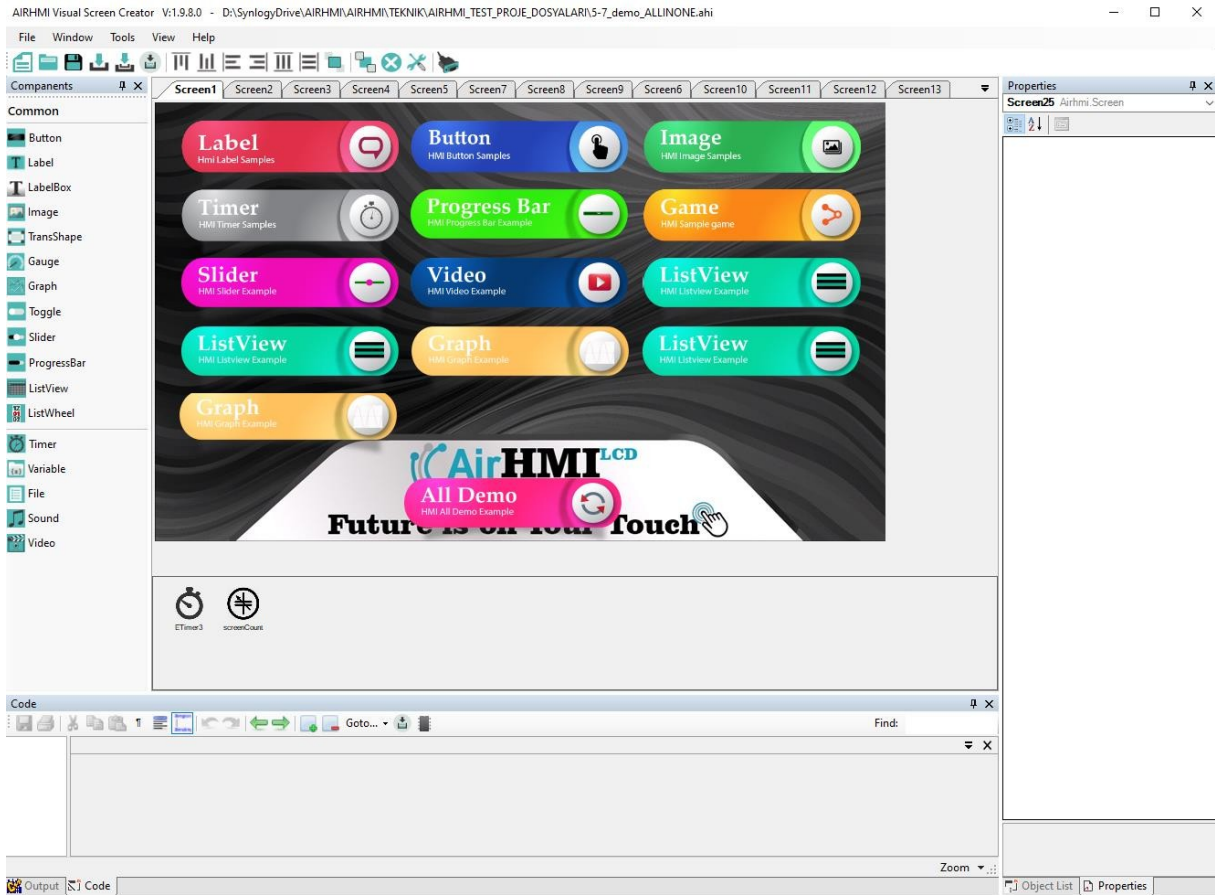
1) The pins of the POWER connector are as follows;



Warning: Do not reverse the 5V supply. If you reverse the supply, your screen may be damaged.

AIRHMI LCD SCREEN EDITOR MANUAL

4. AirHMI EDITOR MAIN INTERFACE



4.1 TITLE BAR

The Title Bar contains the application name and version number when an AirHMI project is opened.

4.2 MAIN MENU and TOOL BARS



AIRHMI LCD SCREEN EDITOR MANUAL

File Menu

For users, there are commands such as Open New Project, Save Project, Save Project As, Open an Existing Project and Exit. The important point here is that when you want to open a new project while an existing project is open, if you want to keep the old project on the computer or if you want the changes made not to be lost, the save message on the screen should be approved.

Window

Within the window area;

- Create a new work screen in addition to the main screen used in the project (Add Screen)
- Downloading the designed interface screen to the AirHMI LCD Board via the selected USB port (Download to Flash)
- The designed interface screen can be extracted as external files to a desired file in the computer (Download to SD Card). It is used for Bootloader loading via SD Card when USB loading is not desired. When the files are copied to the SD Card and the project is run on the SD Card, the files are loaded from the SD Card as if they were loaded via USB.

Tools

In Options in Tools, there is a section for selecting the port for USB upload and setting the baud rate. Since USB upload works at many baud rates, the user can select the desired baud rate setting and perform the upload.

Alignment



AIRHMI LCD SCREEN EDITOR MANUAL

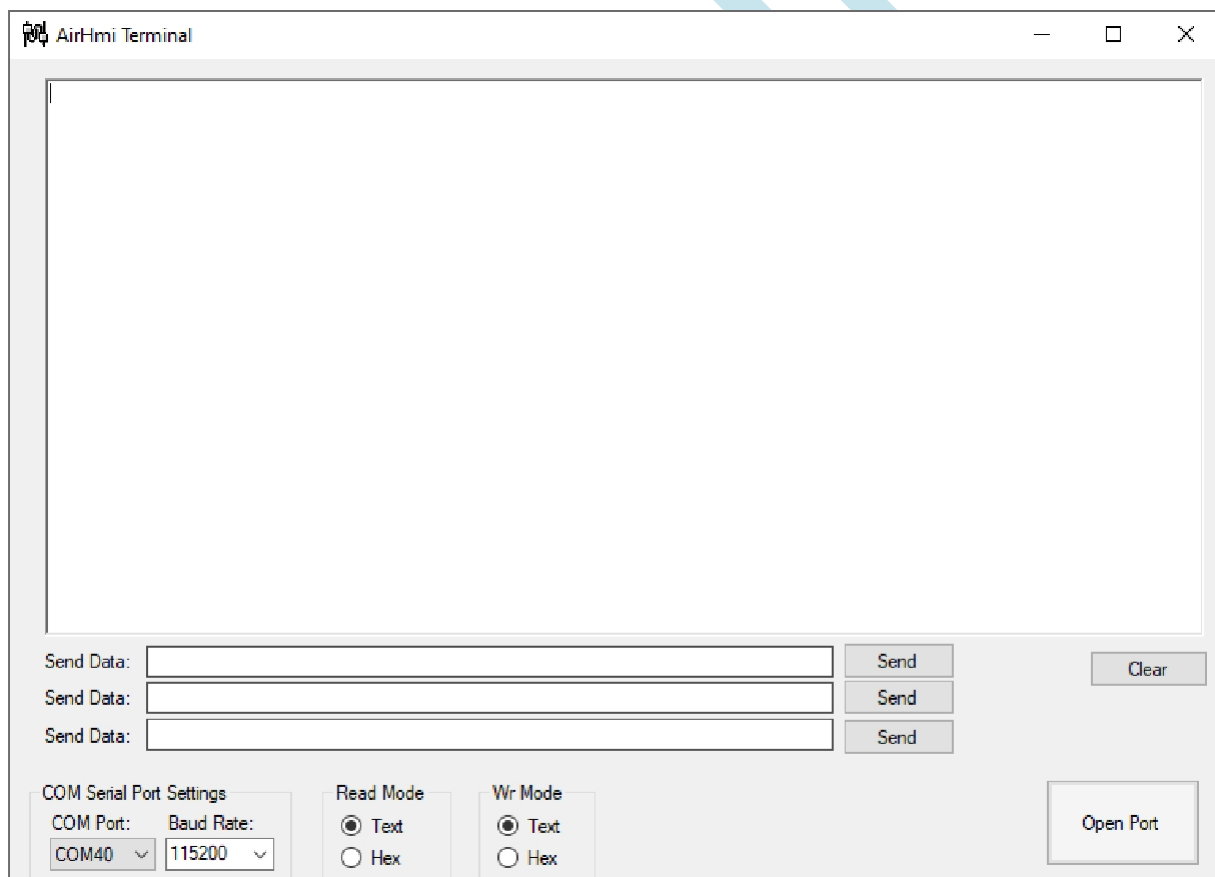
Left Align, Right Align, Top Align and Bottom Align; vertically and horizontally averaging features align or center the specified objects as desired.

Bring to Front and Send to Back properties can be used to determine which of the nested objects will be in the front and used for objects that are desired to be in the background.

Serial Port Terminal Screen

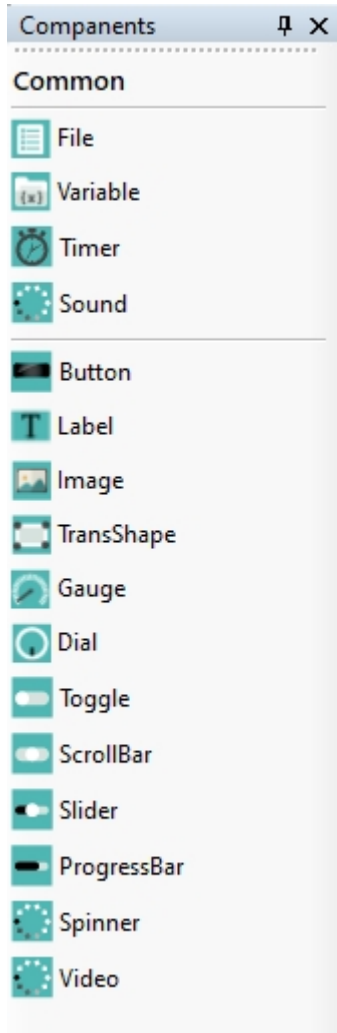


When you click on the serial port icon in the Tools window,



This window opens. With your Airhmi screen

4.3 COMPONENTS COMPARTMENT



Ready-made objects to be shown on the AirHMI LCD Design Screen found in is the section. To be used

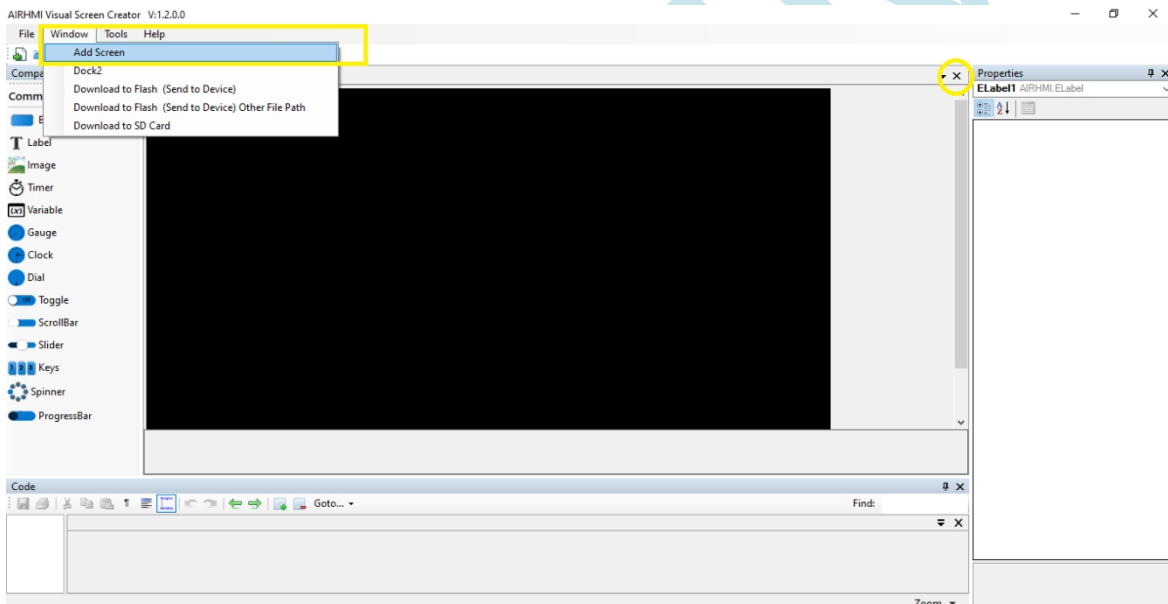
The desired object can be added to the project by clicking on it and dragging it to the screen area. External objects that are not shown on the screen are also in this section: Timer and Variable. These objects are located at the bottom of the screen area in the Non-Visual Components Area section. Setting the properties of objects (position, size, name, etc...) specific to the designed project is available in the section called Objects Attribute Area.

AIRHMI LCD SCREEN EDITOR MANUAL

4.4 SCREEN / COMMAND TAB

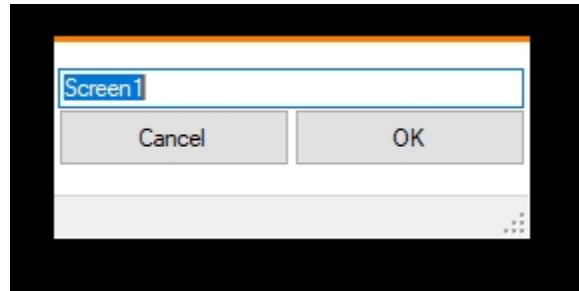


Design projects are generally not used as a single screen but need different screens at the same time. Opening General Display Screen, Menu Setting Screen, Detailed Display Screen, etc... For this reason, the user can design more than one original and creative screen in the AirHMI Editor in line with their requests. With the Screen / Command Tab, the process of selecting which screen to work on is realized.



To add a new worksheet, you can use the Window/Add Screen tab or right-click on an empty space on the worksheet and select Add Screen. To delete the opened worksheet, just press the cross (x) at the end of the Screen / Command Tab line.

To change the name of the screen, right click on an empty area on the screen and click on the Rename tab. The name of the screen can be changed in the tab that opens.



4.5 DESIGN MAIN SCREEN AREA

AIR HMI Designer work screen design image area. In LCD screen design, features such as which objects will be located where on the screen, their sizes, text features are shown in this area.

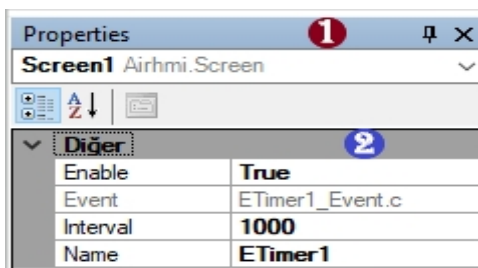
AHMI SCREEN EDITOR

4.6 AREA OF COMPONENTS WITHOUT VISUALS



Not all components in a project are shown on the LCD screen. There are also components that do not need to be shown on the LCD screen while performing very important tasks in the background: Timer and Variable. It is important to show the components that are not shown on the LCD screen but work in the background in the Editor for ease of use and understandability during design. The Non-Visual Components Area is the area where components such as Timer and Variable used in the project are shown.

4.7 ATTRIBUTE SPACE OF OBJECTS



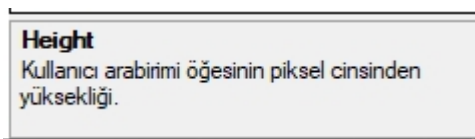
4.83.7.1 Display Area of Objects Used in the Project

Many objects can be used in LCD screen design. Each object has its own settings. In projects where more detail is required, it can be complicated to find the object to be adjusted from the design screen. In order to prevent this complexity, this is the area where the list of all objects used in the design is located. In this way, the desired object can be selected and its settings can be made in the Attribute area.

4.93.7.2 Attributes of Objects Display / Setting Area

In AirHMI Editor, objects are automatically added with their initial settings when they are included in the project. Users can edit many properties of the objects they add according to their usage purposes and requests such as names, sizes, appearance, colors, etc. in this area.

4.10 ATTRIBUTES DESCRIPTION FIELD



The settings of the objects are performed in the attribute field. But only the attribute name is written there. In the Attribute Description field, there is a description of the attributes. The functions that the attribute headers fulfill are generally explained.

4.11 USER PROJECT CODE MENU and TOOL BARS



The most important part of the designed project is the code phase. According to the basis of the project, what will be shown on the design screen in which situations is set by the coding structure. The Code Menu contains some basic components that can help the user in writing code, such as save code, copy and paste, search for keywords in the code and so on.

4.12 USER PROJECT CODE FIELD



AIRHMI LCD SCREEN EDITOR

It contains a valid AirHMI PICOC Code Instruction for User Project Code. This section will not teach programming, but will generally help the user to add code. In this area, users will be able to write C-based codes to the events of the Timer component or the events of the objects they use on the screen. Thanks to the ready-made library codes supported by Screen Editor, the software difficulty is minimized.

You can examine the ready-made functions for the section in detail under the third heading (3. Functions). In addition to the functions mentioned there, all of the C-based code is included in this field.

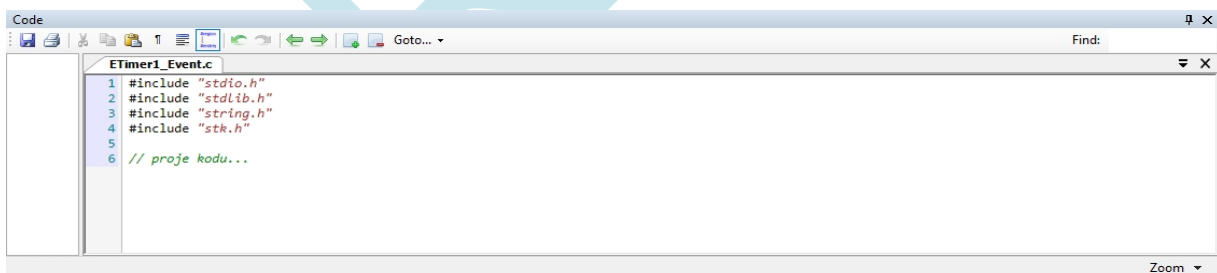
can be written and run simultaneously in the program.

4.13 CODE AREA ZOOM AREA



In the project design, in order to make the code field text size easier for the user to use is the area where you can zoom in and out to the desired extent.

4. 14CODE FIELD

A screenshot of a code editor window titled 'Code'. The window has a standard toolbar with icons for file operations and editing. The main area shows a C code file named 'ETimer1_Event.c'. The code is as follows:

```
1 #include "stdio.h"
2 #include "stdlib.h"
3 #include "string.h"
4 #include "stk.h"
5
6 // proje kodu...
```

The text is color-coded: preprocessor directives are in blue, and comments are in green. A 'Zoom' dropdown menu is visible in the bottom right corner of the editor window.

In addition to the solution-oriented structure of AirHMI Editor, which aims to create a design at the most efficient point in terms of time and effort, one of the most important advantages is its easy and understandable code structure. The code structure is prepared in C programming language. However, in order to be user-oriented and to provide ease of use to the user, the necessary functions are prepared under the "stk.h" library. In this layout where basic C libraries are attached, you can create your code using the C programming language and add the necessary functions to your code.

AIRHMI LCD SCREEN EDITOR MANUAL

you can add per C function. In addition to the ready-made C functions, you can find ready-made functions with explanations on many important topics such as object control / setting functions, LCD screen sleep mode, timer code layout. The important point here is that the "stk.h" library must be added to the beginning of each code structure for these functions to work actively.

```
ETimer1_Event.c
1 #include "stdio.h"
2 #include "stk.h"
3
4 char uartData[10];
5 int uartsz;
6 uartDataGet(uartData, &uartsz);
7
8 if(uartsz > 0)
9 {
10     ImageSet ("EImage1" , "Visible" , "1");
11     LabelSet ("ELabel1" , "Caption" , "Deneme");
12     LocalIntVarSet("Variable1" , 2);
13
14     DrawScreenGet();
15
16 }
```

The sample code structure is prepared with timer. Detailed explanation for timer code structure **2.1 TIMER**

is described under the heading.

The code structure can be in Timer according to the desired situation, as well as the code structure that we want to run when objects are touched for resistive screens. The code you will create in the Event in the Timer should be added to the OnUp section in the attribute section in the same way as the code structure you want to be active when the objects are touched while the timer interval is active in the entire program.

5. Transformations of Variables into Each Other

Airhmi displays use the C programming infrastructure. In order to perform operations in the functions below, we will often need type conversions. Type conversions are necessary for operations between mathematical expressions and data. In addition, data must be converted to the same type as the defined variable in order to save the data in variable objects. This is very important in order to use the Airhmi screen effectively. Detailed examples will give detailed information about all the necessary transformations. Standard C codes `sprintf`, `atoi` and `aof` can be used for the conversion of variables. In addition, airhmi offers you ready-made functions to make operations easier. You can do type conversions using the following functions.

5.1 Converting an Integer Expression to String (Char Array)

In Airhmi, integer expressions are variables that take integer values and occupy 4

```
bytes. int i; i=5;
```

There may be a need to convert the integer `i` expression here to show it on a label in airhmi or to send it as data to a field with `char *`. In this case you can use the `Convert_IntToString` function.

```
void Convert_IntToString(int,char *)
```

Used to convert integer variables to `char *` array. Example

Usage:

```
#include "stk.h"  
int i = 5;  
char data[200];
```

```
Convert_IntToString(i,data);  
LabelSet("Label1", "Text",data);
```

In this example, the `LabelSet` function accepts a variable of type `char *` as parameter 3.

AIRHMI LCD SCREEN EDITOR MANUAL

We cannot give the variable `i` directly to this function. Therefore, with the `Convert_IntToString` function, we have converted it to the `char *` type required for the `LabelSet` function.

void Convert_FloatToString(float,char *)

The use of float expressions is quite common in Airhmi. To show float expressions in a Label or to use them as `char *` in another function, a conversion is required.

Example Usage:

```
#include "stk.h"
float i = 5.2;
char data[200];

Convert_FloatToString(i,data);
LabelSet("Label1", "Text",data);
```

void Convert_StringToInt(char *,int *)

Converting String(`char *`) expressions to integers is often used to perform mathematical operations with a text. For example, let's take a label value, multiply it by 2 and write it to another label.

```
#include "stk.h"
int i;
char data[200];

LabelGet("Label1", "Text",data);
Convert_StringToInt(data,&i);
i = i * 2;
Convert_IntToString(i,data);
LabelSet("Label2", "Text",data);
```


AIRHMI LCD SCREEN EDITOR MANUAL

```
void Convert_StringToFloat(char *,float *)
```

We may need to subject a string comma expression to mathematical processing. A value entered from the keyboard needs to be converted to a flat expression for operations such as multiplication and division.

```
#include "stk.h"
float i;
char data[200];

LabelGet("Label1", "Text",data);
Convert_StringToFloat(data,&i);
i = i * 2.5;
Convert_FloatToInt(i,data);
LabelSet("Label2", "Text",data);
```

* The reason for float * as the 2nd parameter in the Convert_StringToFloat function is that the function fills the value of the float variable i and gives it back to us. Actually pointer is used here. If you do not know about pointers, you can briefly search the internet.

5.2 Using sprintf

sprintf is used for many functions in c, such as type conversions or concatenating

expressions. Example 1:

```
char data[20];
int i = 5;
sprintf(data,"%d",i); // we have converted the integer value i to char *
```

Example 2:

```
char data[20];
int i = 5;
int k = 6;
sprintf(data,"%d%d",i,k); // We wrote i and k in a single variable as char *. The
content of data = 56.
```

Example 3:

```
char data[20];
float i = 5.2;
int k = 6;
sprintf(data,"%f%d",i,k); // We wrote i and k as char * in a single variable. data's
content = 5.26.
```

There are plenty of examples on the internet for other uses of sprintf.

5.3 atoi

atoi is a function in the C programming language and is used to convert a string to an integer. Typically, strings are used when processing data received from places such as user input or file reads.

A simple use case of the atoi function:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    char str[] = "12345";
    int number = atoi(str);
    return 0;
}
```

In this example, the string "12345" is passed to the atoi function, which converts it to an integer. As a result, the number variable becomes 12345. You can now perform operations on the number variable as an integer.

5.4 atof

The atof function is another commonly used function in the C programming language. This function is used to convert a string into a decimal number.

A simple use case of the atof function: #include

```
<stdio.h>
#include <stdlib.h>

int main() {
    char str[] = "3.14";
    double number = atof(str);
    return 0;
}
```

In this example, the string "3.14" is passed to the atof function, which converts it to a decimal number. As a result, the number variable becomes 3.14.

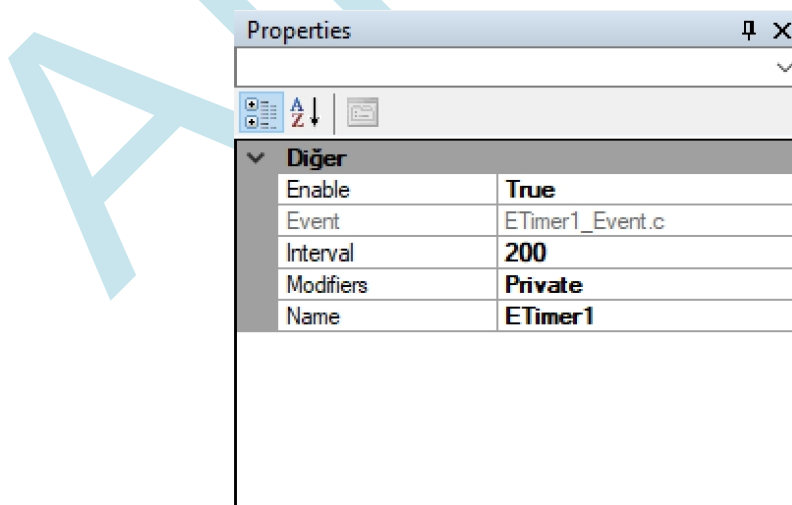
6. ARHMIC OBJECTS AND FUNCTIONS

6.1 TIMER

Perhaps the most important point in the code structure is the use of Timer. The changes that will occur in the real-time operation of the designed editor screen in the project and the intervals at which these changes will occur are set in the Timer Attributes. Enable selects whether the Timer will be active or not. Interval selects the intervals in milliseconds at which the code will be active. Name, as the name suggests, is the name of the Timer. Event section is the section to open the code section to be created for the project design. ETimer1_Event.c is the name of the C file where the generated code is saved.

In Timer usage, the code structure activates the code directory at those intervals according to the time set in the Interval, regardless of the state of the objects. If the user uses a resistive screen in his project and wants to perform an operation when an object is touched; he needs to come to the OnUp section of the object he wants to perform an operation when touched and add the code under this attribute. Thus, the code written only when that object is touched will be active regardless of the Timer.

Timer Properties Window



A I R H M I LCD SCREEN EDITOR

Feature	Option	Description
Enable	True False	Enables the timer object. Makes the timer object disable.
Name		This is the name of the object used for design. This name is used in the object name section of the code.
Event		Timer object is a software field.
Interval		Sets the timer repeat time.
Modifiers	Private Public	It is the timer that works only on this page. Timer that works on all pages.

Functions

1. TimerSet ()

Description

It is the command that regulates the parameter settings of the button object.

Function

```
void TimerSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

AIRHMI LCD SCREEN EDITOR MANUAL

Enable command

TimerSet(Object name , "Enable" , "1 , 0 or True , False");

Sample Code:

```
ButtonSet ("Timer1" , "Enable" , "True");
```

Interval command

TimerSet(Object name , "Interval" , "Value in milliseconds."); Sample

Code:

```
ButtonSet ("Timer1" , "Interval" , "1000"); // sets the interval to 1 second.
```

AIRHMI

6.2 Button

A button object is an object that allows any action to be taken when it is pressed. For example, it can be used to send the data received from the user somewhere, to perform an operation with the received data or to give a message. You can drag the position of the button wherever you want and adjust its size by pulling its edges.

Button Shapes



AIRHMI LCD SCREEN EDITOR MANUAL

Button Properties Window

Properties 🔍 ✕

EButton11 AIRHMI.EButton ▼

🔍 🔍 🔍 🔍

Diğer	
Active	False
Caption	EButton 1
Color	■ Red
ColorTo	□ White
Name	EButton 11
OnDown	
OnPress	
OnUp	EButton11_OnUp.c
Pen Color	■ Black
Pen Width	1
PressColor	□ White
PressColorTo	■ Silver
Static	False
Visibled	True

Görünüm	
TextAlign	MiddleCenter

Others	
Gradient	None

Yazı Tipi	
Font Color	□ White
Font Name	Roboto
Font Size	8

Yerleşim	
Dock	None
Height	60
Left	100
Top	100
Width	120

A I R H M I LCD SCREEN EDITOR

Feature	Option	Description
Active	True False	Allows the button object to be pressed. The button object does not allow the press function.
Caption,Text		The name of the button object on the screen.
Color		Specifies the color of the button object on the screen.
ColorTo		If the Gradient property is selected, a transitional button object is created on the screen. It is used to define the transition color of this object from Color to ColorTo.
Name		The name of the object used for design. In the code section this name is used in the object name section.
OnDown		The piece of code that runs during the button object press function is written here.
OnPress		It will work as long as we keep our hand pressed on the button object is the piece of code. It works repetitively.
OnUp		Code that runs when we withdraw our hand from the button object part is written here.
Border Color		It is the color to indicate the boundaries around the Button Object in the form of a line.
Border Color		The line created around the button object thickness.
Press Color		Specifies the color of the button object on the screen when pressed.
Press ColorTo		If the Gradient property is selected, while pressed, a button object with a transition is created on the screen. To define the transition color of this object from Press Color to Press ColorTo used.
Static		
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Text Aling		It is the positioning of the text on the button object relative to the button.
Gradient	None Top to Buttom Left to Right	The Gradient feature is turned off. ColorTo and Press ColorTo are disabled. Gradient colors are applied from top to bottom. Gradient colors are applied from left to right.
Font Color		The text color of the button.
Font Name		Defining different font options for the button object It is done.
Font Size		The size of the font of the object's text.

A I R H M I LCD SCREEN EDITOR

Dock	MANUAL	The way the button object is snapped to the screen. Full screen You can make a laying process.
Height		It is the height of the object.
Left		Specifies the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate
Width		It is the width of the object.

Functions

2. ButtonSet ()

Description

It is the command that regulates the parameter settings of the button object.

Function

```
void ButtonSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Visible adjustment command

```
ButtonSet( Object name , "Visible" , "1 , 0 or True , False" );
```

When the Value property is set to "True" the button object appears, when it is set to "False" it does not appear.

A I R H M I LCD SCREEN EDITOR

Sample Code: MANUAL
ButtonSet ("EButton1" , "Visible" , "True");

Active setting command

ButtonSet(Object name , "Active" , "1 , 0 or True , False");

AIRHMI LCD SCREEN EDITOR MANUAL

Sample Code:

```
ButtonSet("EButton1" , "Active" , "True");
```

Left adjustment command

```
ButtonSet( Object name , "Left" , "X coordinate" );
```

Sample Code:

```
ButtonSet("EButton1" , "Left" , "10");
```

Ball adjustment command

```
ButtonSet( Object name , "Top" , "Y coordinate" );
```

Sample Code:

```
ButtonSet("EButton1" , "Top" , "255");
```

Width adjustment command

```
ButtonSet( Object name , "Width" , "Size ( 0 to Screen X size)" ); Sample
```

Code:

```
ButtonSet("EButton1" , "Width" , "90");
```

Height adjustment command

```
ButtonSet( Object name , "Height" , "Size (from 0 to Screen Y size)" );
```

Sample Code:

```
ButtonSet("EButton1" , "Height" , "70");
```

Color adjustment command

```
ButtonSet( Object name , "Color" , "RGB Color in hex format #RRGGBB" );
```

Sample Code:

```
ButtonSet("EButton1" , "Color" , "#FFA07A");
```

ColorTo adjustment command

```
ButtonSet( Object name , "Color To" , "RGB Color in hex format #RRGGBB" );
```

Sample Code:

```
ButtonSet("EButton1" , "ColorTo" , "#FFA07A");
```

AIRHMI LCD SCREEN EDITOR MANUAL

Press_Color setting command

ButtonSet(Object name , "Press Color" , "RGB Color in hex format #RRGGBB");

Sample Code:

```
ButtonSet("EButton1" , "Press_Color" , "#FFA07A");
```

Press_ColorTo adjustment command

ButtonSet(Object name , "Press ColorTo" , "RGB Color in hex format #RRGGBB"

); Sample Code:

```
ButtonSet("EButton1" , "Press_ColorTo" , "#FFA07A");
```

FontSize adjustment command

ButtonSet(Object name , "FontSize" , "Font size is set between 8-102."); Sample

Code:

```
ButtonSet("EButton1" , "FontSize" , "12");
```

Font_Color setting command

ButtonSet(Object name , "Font Color" , "RGB Color in hex format #RRGGBB");

Sample Code:

```
ButtonSet("EButton1" , "Font_Color" , "#FFA07A");
```

Caption setting command

The string expression of the button object that appears on the screen is

changed with this command. ButtonSet(Object name , "Caption and Text" ,

"Hello World!");

Sample Code:

```
ButtonSet("EButton1" , "Caption" , "Hello World!");
```

```
ButtonSet("EButton1" , "Text" , "Hello World!");
```

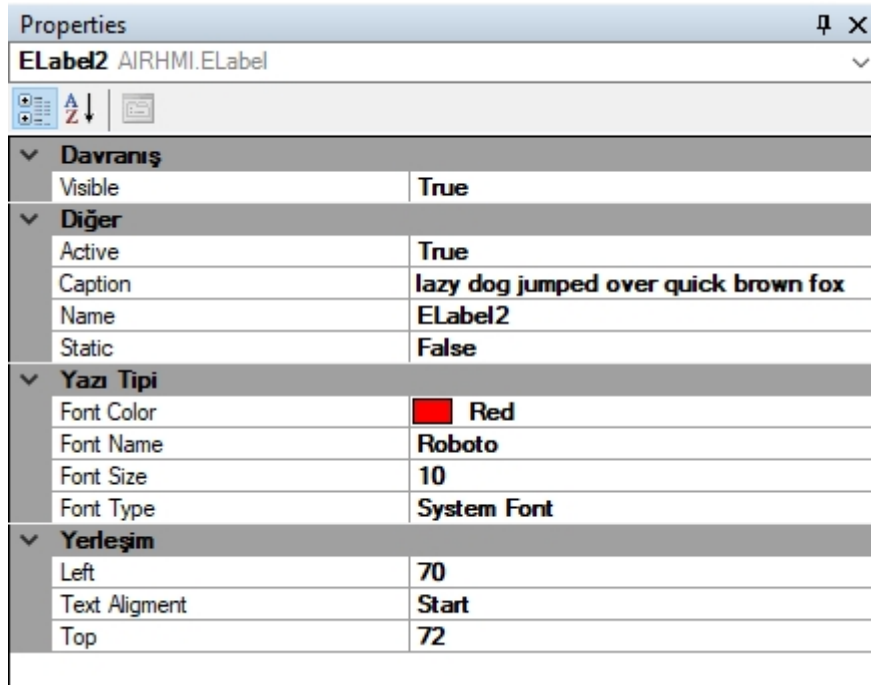
6.3 Label

It is an object used for writing on the screen. It supports font size from 8 to 102. Default Font is "Roboto".



AIRHMI LCD SCREEN EDITOR MANUAL

Label Properties Window



Feature	Option	Description
Active	True False	If turned on, the keyboard will automatically appear when the label is touched. The keyboard is inactive.
Caption ,Text		The text of the Label object that appears on the screen.
Color		Specifies the color of the button object on the screen.
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Name		It is the name of the object used for design. This name is used in the object name section of the code.
Static		Reserved.
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Text Alingment	Start Center	Label object left justification, Label object average
Font Color		Label's text color.
Font Name		Different font options are defined for the Label object.
Font Size		The size of the font of the object's text.
Height		It is the height of the object.
Left		Specifies the position on the screen. X coordinate

A I R H M I LCD SCREEN EDITOR

Top	MANUAL	Indicates the position on the screen. Y coordinate
Width		It is the width of the object.

Functions

LabelSet ()

Description

It is the command that regulates the parameter settings of the Label object.

```
void LabelSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Active setting command

```
LabelSet( Object name , "Active" , "1 , 0 or True , False" );
```

Sample Code:

```
LabelSet("ELabell" , "Active" , "True");
```

Visible adjustment command

```
LabelSet( Object name , "Visible" , "1 , 0 or True , False" );
```

Sample Code:

```
LabelSet("ELabell" , "Visible" , "1");
```


AIRHMI LCD SCREEN EDITOR MANUAL

Left adjustment command

LabelSet(Object name , "Left" , "10");

Sample Code:

LabelSet(*"ELabell"* , *"Left"* , *"10"*);

Ball adjustment command

LabelSet(Object name , "Top" , "255");

Sample Code:

LabelSet (*"ELabell"* , *"Top"* , *"255"*);

FontSize adjustment command

LabelSet(Object name , "FontSize" , "16"

); Sample Code:

LabelSet(*"ELabell"* , *"FontSize"* , *"16"*);

Font_Color setting command

LabelSet (Object name , "Font_Color" , "RGB Color in hex format #RRGGBB");

Sample Code:

LabelSet(*"ELabell"* , *"Font_Color"* , *"#FFA07A"*);

Caption, Text setting command

The string expression of the Label object that appears on the screen is changed

with this command. LabelSet (Object name , "Caption and Text" , "Hello

World!");

LabelSet (*"ELabell"* , *"Caption"* , *"Hello World!"*);

LabelSet (*"ELabell"* , *"Text"* , *"Hello World!"*);

AIRHMI LCD SCREEN EDITOR MANUAL

LabelGet()

void **LabelGet**(unsigned char *name , unsigned char *type , unsigned char *value)

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Caption, Text command

The string expression of the Label object that appears on the screen is changed

with this command. LabelGet (Object name , "Caption and Text" , char *

buffer);

Char value[20];

LabelGet(*"ELabell"* , *"Caption"* , value);

LabelGet(*"ELabell"* , *"Text"* , value);

6.4 Image

Image object can be used to display images and use images as buttons. With the press image property, you can assign two images to an object and change their images in normal state and press state without writing any code.



AIRHMI LCD SCREEN EDITOR MANUAL

Image Properties Window

Properties	
EImage2 AIRHMI.EImage	
Z ↓	
▼ Davranış	
Visible	True
▼ Diğer	
Active	True
Locked	False
Name	EImage2
OnDown	
OnPress	
OnUp	EImage2_OnUp.c
Opacity	100
PictureName	Asset 9.png
PicturePressImage	
ScaleX	0,5935
ScaleY	0,5984
Static	False
▼ Yerleşim	
Dock	None
Height	73
Left	17
Top	242
Width	238

AIRHMI LCD SCREEN EDITOR

Feature	Option	Description
Active	True False	If turned on, the image can be used as a button. If it is off, it is only used as a picture.
Visible	True False	Visible when the screen first appears. Not visible when the screen first appears.
Name		The name of the object used for design. Code this name is used in the object name section.
Static		Reserved.
Locked	True False	Does not allow to change the position of the object placed on the screen. You can move the image to the desired position.
Text Alingment	Start Center	Label object left justification, Label object average
Height		It is the height of the object.
Left		Specifies the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate
Width		It is the width of the object.
Image File		It is the image file you need to upload from your computer.
Press Image File		The image while holding down on the Image object.
ScaleX		The magnification and reduction ratio of the image object in X dimension.
ScaleY		Zoom in and out in Y dimension of an image object is the ratio.
OnDown		The piece of code that runs during the function to print to the Image object is written here.
OnPress		As long as we keep our hand pressed on the Image object is the piece of code that will run. It runs repetitively.
OnUp		This is the code fragment that runs when the Image object is pulled.

Functions

ImageSet ()

Description

It is the command that sets the parameter settings of the Image object.

Function

```
void ImageSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Example code

Visible adjustment command

```
ImageSet( Object name , "Visible" , "1 , 0 or True , False" );
```

Sample Code:

```
ImageSet("EImage1" , "Visible" , "True");
```

Left adjustment command

```
ImageSet( Object name , "Left" , "Left Position" );
```

Sample Code:

```
ImageSet ("EImage1" , "Left" , "10");
```

AIRHMI LCD SCREEN EDITOR MANUAL

Ball adjustment command

ImageSet(Object name , "Ball" , "Ball Position");

Sample Code:

ImageSet (*"EImage1"* , *"Top"* , *"255"*);

AIRHMI

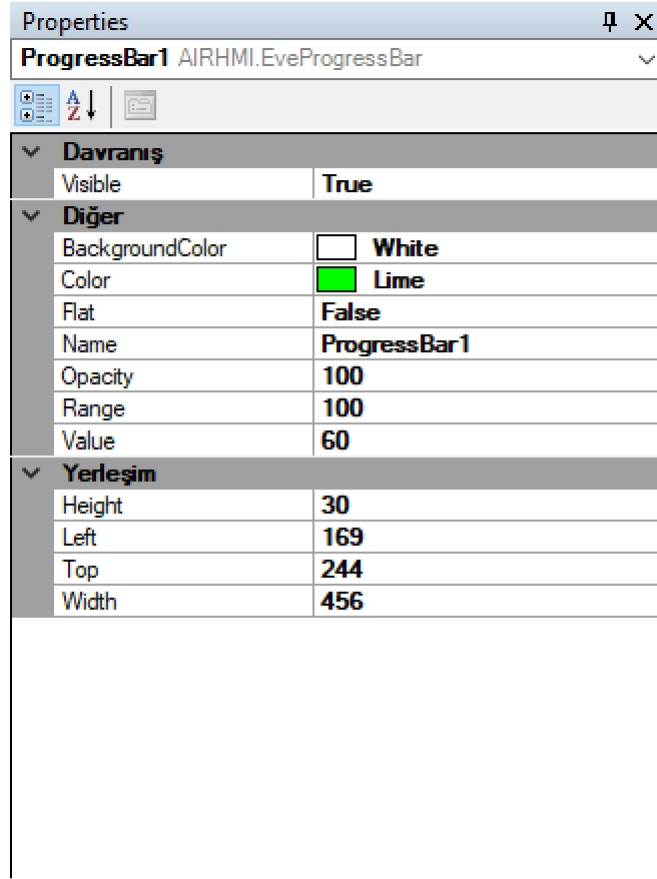
6.5 Progress Bar

Progress Bar means "progress bar" in Turkish. It is used when the execution stages of a long process need to be shown graphically. An example of using Progress Bar: a video or audio file that is being played
time on the Progress Bar, graphical representation of the fill rate of a fuel tank using the Progress Bar.



AIRHMI LCD SCREEN EDITOR MANUAL

ProgressBar Properties Window



Properties	
ProgressBar1 AIRHMI.EveProgressBar	
▼ Davranış	
Visible	True
▼ Diğer	
BackgroundColor	White
Color	Lime
Flat	False
Name	ProgressBar1
Opacity	100
Range	100
Value	60
▼ Yerleşim	
Height	30
Left	169
Top	244
Width	456

AIRHMI LCD SCREEN EDITOR

Feature	Option	Description
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Name		This is the name of the object used for design. This name is used in the object name section of the code.
Color		The middle part of the progressbar object indicates the color.
BackgroundColor		Specifies the background color of the Progressbar object.
Range		Progress bar specifies how many values there will be in total.
Value		Specifies what percentage the progressbar will start at when it is loaded on the first screen.
Height		It is the height of the object.
Left		Specifies the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate
Width		It is the width of the object.

AIRHMI

Functions

ProgressBarSet ()

Description

It is the command that regulates the parameter settings of the Progress Bar object.

Function

```
void ProgressBarSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Example code

Visible adjustment command

```
ProgressBarSet( Object name , "Visible" , "1 , 0 or True , False" );
```

Sample Code:

```
ProgressBarSet("ProgressBar1" , "Visible" , "False");
```

Left adjustment command

```
ProgressBarSet( Object name , "Left" , "X coordinate position on the screen"
```

); Sample Code:

```
ProgressBarSet("ProgressBar1" , "Left" , "10");
```

Ball adjustment command

ProgressBarSet(Object name , "Top" , "Y coordinate position on the screen"

); Sample Code:

```
ProgressBarSet("ProgressBar1" , "Top" , "255");
```

Color adjustment command

ProgressBarSet(Object name , "Color" , "RGB Color in hex format #RRGGBB");

Sample Code:

```
ProgressBarSet("ProgressBar1" , "Color" , "255");
```

BackGround_Color setting command

ProgressBarSet(Object name , "BackGround_Color" , "RGB Color in hex format #RRGGBB");

Sample Code:

```
ProgressBarSet("ProgressBar1" , "BackGround_Color" , "1458269");
```

Range adjustment command

ProgressBarSet(Object name , "Range" , "Range (numeric)");

Sample Code:

```
ProgressBarSet("ProgressBar1" , "Range" , "100");
```

Value setting command

ProgressBarSet(Object name , "Value" , "Value (numeric)");

Sample Code:

```
ProgressBarSet("ProgressBar1" , "Value" , "50");
```

6.6 Slider

A slider or trackbar is a graphical control element where the user can set a value by moving an indicator horizontally or vertically. In some cases, the user can also click a point on the slider to change the setting.



AIRHMI LCD SCREEN EDITOR MANUAL

Slider Properties Window

Davranış	
Visible	True
Diğer	
Active	True
BackgroundColor	DeepSkyBlue
Color	Gray
direction	vertical
Flat	False
Name	Slider1
OnDown	
OnUp	
Opacity	255
PressColor	128; 255; 128
Range	100
ThumbColor	Lavender
Value	50
Yerleşim	
Height	181
Left	196
Top	62
Width	56

AIR HMI LCD SCREEN EDITOR

Feature	MA Option	Description
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Active	True False	Allows press function on the slider object. The slider object does not allow the press function.
Name		The name of the object used for design. Code this name is used in the object name section.
Color		The color of the back part of the slider object.
BackgroundColor		Specifies the background color of the slider object.
ThumpColor		The color of the round part on the slider object.
PressColor		When the slider object is pressed, the round changes the color of the part.
Range		Progress bar specifies how many values there will be in total.
Value		Specifies what percentage the progressbar will start at when it is loaded on the first screen.
Direction		Control the Vertical, Horizontal Slider object on the screen direction.
Height		It is the height of the object.
Left		Specifies the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate
Width		It is the width of the object.

AIR

AIRHMI LCD SCREEN EDITOR MANUAL

SliderSet ()

Description

It is the command that regulates the parameter settings of the slider object.

Function

```
void SliderSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Visible adjustment command

```
SliderSet( Object name , "Visible" , "1 , 0 or True , False" );
```

Sample Code:

```
SliderSet("Slider1" , "Visible" , "1");
```

Left adjustment command

```
SliderSet( Object name , "Left" , "X coordinate position on the  
screen" );
```

Sample Code:

```
SliderSet("Slider1" , "Left" , "10");
```

Ball adjustment command

```
SliderSet( Object name , "Top" , "Y coordinate position on the  
screen" );
```

Sample Code:

```
SliderSet("Slider1" , "Top" , "255");
```


AIRHMI LCD SCREEN EDITOR MANUAL

SliderGet ()

Description

It is the command to get the parameter settings of the slider object.

Function

```
void SliderGet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Value command

```
SliderGet( Object name , "Value" , "char * buffer"
```

```
); Sample Code:
```

```
char buffer[20];
```

```
SliderGet("Slider1" , "Value" , buffer);
```

6.7 Gauge

The Gauge object is an effective object for displaying analog values. It is also used as a speedometer.



AIRHMI LCD SCREEN EDITOR MANUAL

Gauge Properties Window

Properties	
Gauge1 AIRHMI.EveGauge	
A ↓ Z ↓	
▼ Davranış	
Visible	True
▼ Diğer	
Active	True
Color	Blue
Flat	False
MajorCount	10
MinorCount	5
Name	Gauge1
OnDown	
OnUp	
PenColor	Red
PressColor	White
Radius	94
Range	100
Tag	255
TicksVisible	False
Value	0
▼ Yerleşim	
Left	59
Top	39

A I R H M I LCD SCREEN EDITOR

Feature	MA Option	Description
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Name		It is the name of the object used for design. This name is used in the object name section of the code.
Color		The portion of the back of the gauge object is the color.
BackgroundColor		Specifies the background color of the slider object.
PressColor		When the slider object is pressed, the round changes the color of the part.
Range		Progress bar specifies how many values there will be in total.
Value		From what percentage when the progressbar is loaded on the first screen will start.
Radius		Sets the diameter of the gauge object.
TicksVisible		Turns the lines around the Gauge object on and off.
Left		Specifies the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate

Functions

GaugeSet ()

Description

This command sets the parameter settings of the Gauge object.

Function

```
void GaugeSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Example code

Visible adjustment command

```
GaugeSet( Object name , "Visible" , "1 , 0 or True , False" );
```

Sample Code:

```
GaugeSet("Gauge1" , "Visible" , "1");
```

Left adjustment command

```
GaugeSet( Object name , "Left" , "X coordinate position on the screen" );
```

Sample Code:

```
GaugeSet("Gauge1" , "Left" , "10");
```

Ball adjustment command

GaugeSet(Object name , "Top" , "Y coordinate position on the screen");

Sample Code:

```
GaugeSet("Gauge1" , "Top" , "255");
```

Color adjustment command

GaugeSet(Object name , "BackGround_Color" , "RGB Color in hex format #RRGGBB");

Sample Code:

```
GaugeSet("Gauge1" , "Color" , "#ffaa02");
```

Value setting command

GaugeSet(Object name , "Value" , "Value (numeric)");

Sample Code:

```
GaugeSet("Gauge1" , "Value" , "100");
```

Range adjustment command

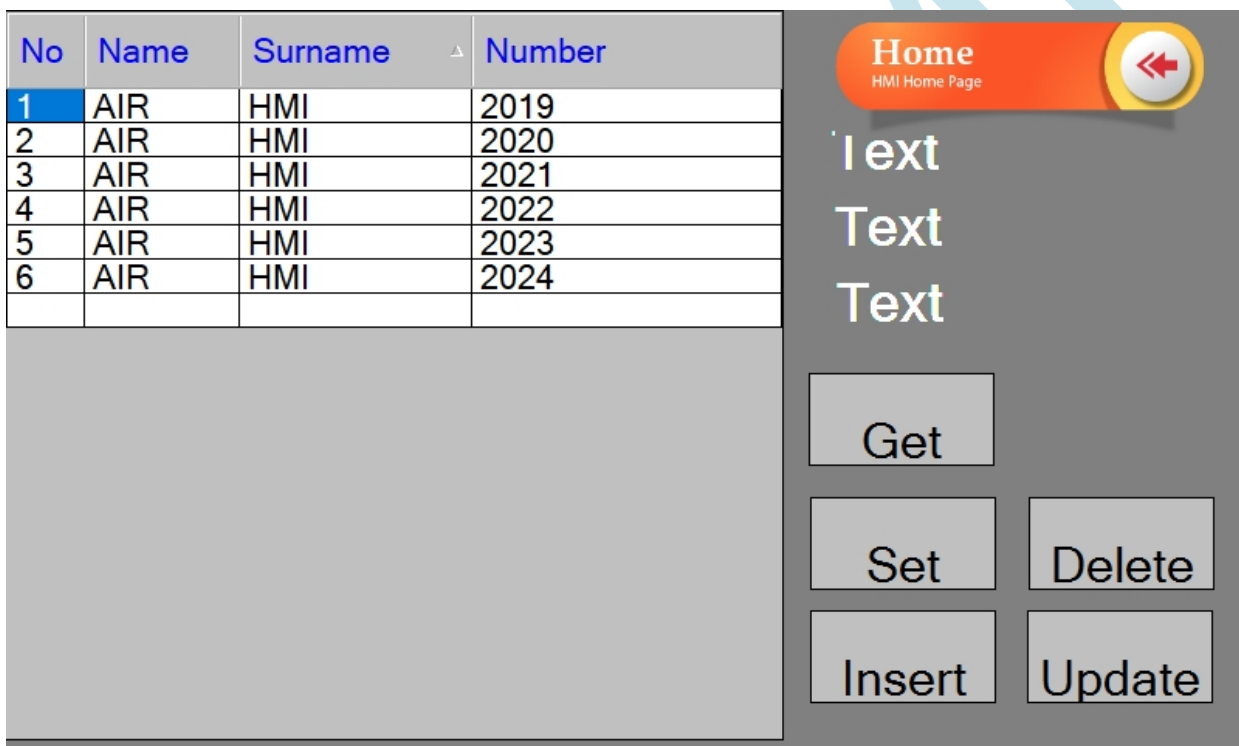
GaugeSet(Object name , "Range" , "Value (numeric)"

); Sample Code:

```
GaugeSet("Gauge1" , "Range" , "30");
```

6.8 ListView

The ListView object allows users to create a table on the airhmi screen. The data entered in the table can be used in many ways. For example, as a language file, or as a structure where you can keep system settings separately, or as a structure where you can keep system logs. you can use it. You can find many functions such as adding data to the list, updating, deleting and reading data below.








The screenshot displays a simulated HMI interface. On the left, a table with four columns: 'No', 'Name', 'Surname', and 'Number'. The first row is highlighted in blue. Below the table is a large grey rectangular area. On the right side of the interface, there is a navigation bar with a 'Home' button (labeled 'HMI Home Page') and a back arrow icon. Below the navigation bar, the text 'Text Text Text' is displayed. At the bottom right, there are five buttons: 'Get', 'Set', 'Delete', 'Insert', and 'Update'.

No	Name	Surname	Number
1	AIR	HMI	2019
2	AIR	HMI	2020
3	AIR	HMI	2021
4	AIR	HMI	2022
5	AIR	HMI	2023
6	AIR	HMI	2024

AIRHMI LCD SCREEN EDITOR MANUAL

ListView Properties Window

List1 AIRHMI.EListView	
✖ Davranış	
Enabled	False
✖ Diğer	
BackGround Color	 Silver
Enable Motion	True
Enable Sort	1
Grid Visible	True
List File	List 1_List File.txt
Locked	False
Name	List 1
Visibled	True
✖ Grid	
Grid Font	Roboto
Grid Font Color	 Black
Grid Font Size	16
selected Item Color	 Blue
✖ Header	
Header BackGround Color	 Silver
Header Font	0
Header Font Color	 Blue
Header Font Size	16
Header Height	50
Header List	No^Name^Surname^Number
Header List Size	50^100^150^200
Header Type	int^str^str^str
✖ Yerleşim	
Height	471
Left	0
Top	0
Width	503

A I R H M I LCD SCREEN EDITOR

Feature	MANUAL Option	Description
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Name		It is the name of the object used for design. This name is used in the object name section of the code.
Left		Indicates the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate
Enable Motion		Allow dragging left and right within the list gives.
Grid Visible		The list is related to the display of grid lines.
List File		Throwing data to the list for the first time from within the editor we can use it whenever we want.
Grid Font		The font of the text in the grid.
Grid Font Color		The color of the text in the grid.
Grid Font Size		The font size of the text in the grid.
Header Font Color		Header Font Color.
Header_BackGround_Color		Header Background Font Color.
Header_Font_Size		Header Font Size dir.
Header_List		The headings section should be written here.
Header_List_Size		Specify the area covered by each heading.
Header_Type		Specifies the type of data.
BackGround_Color		The list is the background.

AIRHMI LCD SCREEN EDITOR MANUAL

ListViewSet()

Description

It is the command that regulates the parameter settings of the ListView object.

Function

```
void ListViewSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Example code

Row value change command

Used to change the value of the place in the selected row. That row must be selected on the touch screen. It starts from Row0 and goes up to Rown. The n value here is the number of columns.

```
ListViewSet( Object name , "Rown" , new value as String );
```

Sample Code:

```
ListViewSet( Object name , "Row0" , "1" );  
ListViewSet( Object name , "Row1" , "AIR" );  
ListViewSet( Object name , "Row2" , "HMI" );
```

Delete_Selected command

AIRHMI LCD SCREEN EDITOR MANUAL

Used to delete the selected line. 3. Parameter does not matter. Therefore 0 We give.

```
ListViewSet( Object name , "Delete_Selected" , 0 );
```

Sample Code:

```
ListViewSet( "List1" , "Delete_Selected" , 0 );
```

update command

It is used to save the list permanently in memory. 3. Importance of the parameter it doesn't exist. That's why we give 0.

```
ListViewSet( Object name , "update" , 0
```

); Sample Code:

```
ListViewSet( "List1" , "update" , 0 );
```

insert command

Used to enter new data into the list. This command adds data to the end of the list.

```
ListViewSet( Object name , "insert" , "data1^data2^data3" );
```

Sample Code:

```
ListViewSet( "List1" , "insert" , "7^air^hmi^2025" );
```

AIRHMI LCD SCREEN EDITOR MANUAL

ListViewGet()

Description

Command to read data from a ListView object.

Function

```
void ListViewGet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Example code

Read row value command

Used to read the value of the place in the selected row. That row must be selected on the touch screen. It starts from Row0 and goes up to Rown. The n value here is the number of columns.
ListViewGet(Object name , "Rown" , value as String);

Sample Code:

```
char row[200]; ListViewGet("List1",  
"Row1",row); ListViewGet("List1", "Row2",row);  
ListViewGet("List1", "Row3",row);
```

AIRHMI LCD SCREEN EDITOR MANUAL

ListViewSetXY()

Description

Used to change the content of the ListView object. It is used like a coordinate system. You do not need to have touched the object to use this function. You can update any field as you like.

Function

```
void ListViewSetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

Parameter	Description
name	Name of the object
X	It is the column number.
Y	Line number.
Value	Sutun value.

Example code

```
ListViewSetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

Sample Code:

```
ListViewSetXY("List1" , 1 , 2 , "new value"); // Row 2 Change the value of column 1.
```

AIRHMI LCD SCREEN EDITOR MANUAL

ListViewGetXY()

Description

Used to read the contents of a ListView object. It is used like a coordinate system. You do not need to have touched the object to use this function. You can read data from any field as you like.

Function

```
void ListViewGetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

Parameter	Description
name	Name of the object
X	It is the column number.
Y	Line number.
Value	Sutun value.

Example code

```
ListViewGetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

Sample Code:

```
char data[100];
```

```
ListViewGetXY("List1" , 1 , 2 , data); // We got the value of row 2, column 1. LabelSet("label1",  
"Text",data);
```

AIRHMI LCD SCREEN EDITOR MANUAL

ListViewSetSort()

Description

Alphabetizes by the first column of the ListView object. This function can be used in the software phase to determine whether the list starts sorted or normal, or to perform this in certain situations. But in order to use it, the "Enable Sort" section must be active in the list properties (Enable Sort = 1). Enable Sort can be enabled by both the editor and the software. You can also use this feature in cases where you want to invert the list, but the first column must have sequential numbers (for example: 1,2,3,4,5).

Function

```
void ListViewSetSort (unsigned char *name ,int value);
```

Parameter	Description
name	Name of the object
Value	1 is used to reverse the object and 0 for the opposite case.

Example code

```
ListViewSetSort(unsigned char *name, int value);
```

Sample Code:

```
ListViewSetSort ("ListView1", 1); // list in reverse form.
```

```
ListViewSetSort ("ListView1", 0); // list in normal form.
```

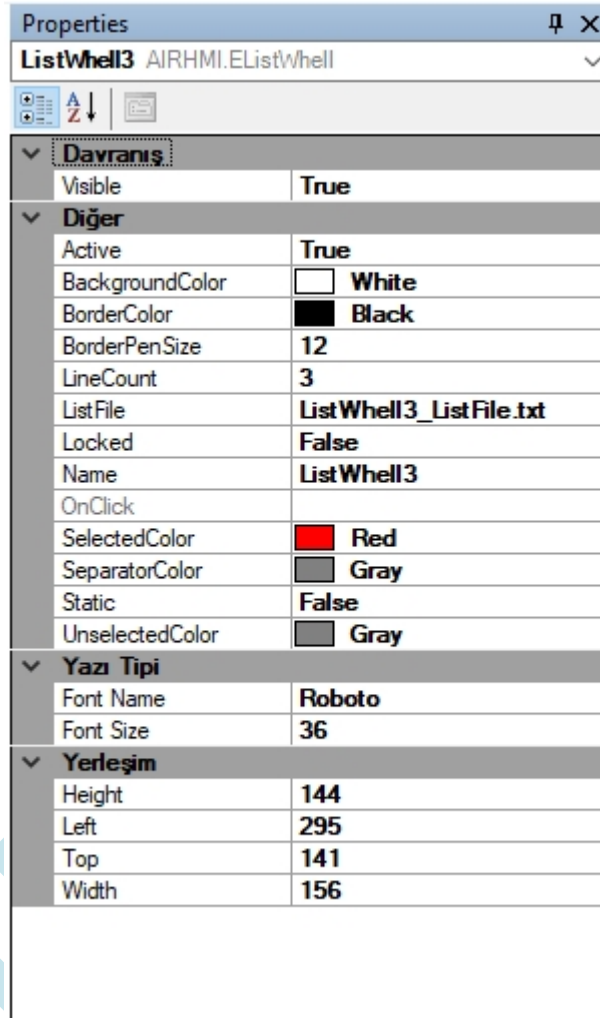
6.9 ListWheel



Allows users to scroll vertically between items in a list. This widget displays each item as a spinning wheel on a cylinder, giving the user a 3D-like scrolling experience. It provides an aesthetic and functional experience to the user, especially in areas such as date pickers and time pickers. ListWheel usually grows as the items of the list come to the center and shrinks as they move away from the center, thus emphasizing the focal point.

AIRHMI LCD SCREEN EDITOR MANUAL

ListWheel Properties Window



Properties	
ListWheel3 AIRHMI.EListWheel	
A Z ↓	
Davranis	
Visible	True
Diğer	
Active	True
BackgroundColor	White
BorderColor	Black
BorderPenSize	12
LineCount	3
ListFile	ListWheel3_ListFile.txt
Locked	False
Name	ListWheel3
OnClick	
SelectedColor	Red
SeparatorColor	Gray
Static	False
UnselectedColor	Gray
Yazı Tipi	
Font Name	Roboto
Font Size	36
Yerleşim	
Height	144
Left	295
Top	141
Width	156

AIRHMI LCD SCREEN EDITOR

Feature	MANUAL Option	Description
Visible	True False	Appears when the screen first appears. The screen is not visible when it first appears.
Name		It is the name of the object used for design. This name is used in the object name section of the code.
Left		Indicates the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate
Line Count		The number of list impressions on the screen.
List File		The numbers that we want to appear on the screen are written one after the other in the list from the editor.
Grid Font		The font of the text in the grid.
Selected Color		It is the color of the selected text.
Unselected Color		The color of other numbers that are not selected.
Sperator Color		It is the color of the line between the numbers.
BackGround_Color		Background color.

AIRHMI LCD SCREEN EDITOR MANUAL

ListWheelSet()

Description

This command sets the parameter settings of the ListWheel object.

Function

```
void ListWheelSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Example code

```
ListWheelSet( Object name , "Value" , new value as String );
```

Sample Code:

```
#include "stk.h"  
char data[20];  
LabelGet("ELabel1" , "Text" , data);  
ListWheelSet("ListWhell1" , "Value",data);
```

AIRHMI LCD SCREEN EDITOR MANUAL

ListWheelGet()

Description

This command sets the parameter settings of the ListWheel object.

Function

```
void ListWheelGet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Example code

```
ListWheelSet( Object name , "Value" , new value as String );
```

Sample Code:

```
#include "stk.h"
```

```
char data[20];
```

```
ListWheelGet("ListWheel1", "Value",data);
```

```
LabelSet("ELabel1" , "Text" , data);
```

AIR HMI LCD SCREEN EDITOR MANUAL

6.10 TransShape

The TransShape object is not visible on the screen but works like a button. When designing, it is placed in the desired position, in the desired size. As a result, you have an invisible button. When clicked, it runs the relevant code block.

Layout	
Height	100
Left	177
Top	241
Width	100
Misc	
Active	True
Locked	False
Name	EShape1
OnDown	
OnPress	
OnUp	

Feature	Option	Description
Height		Height parameter for the entire object (pixel in terms of).
Left		Parameter expressing the position of the object on the horizontal axis (in pixels).
Top		Expresses the position of the object on the vertical axis parameter (in pixels).
Width		Width parameter for the entire object (in pixels).
Active	True False	Allows the function of pressing the TransShape object. Does not allow the function of pressing the TransShape object.
Locked	True False	After moving the object to the desired location on the screen, if the property is <u>True</u> , it will stay there. If the property is <u>False</u> , you can set it to the desired position. It will make your work much easier when you work with multiple objects on the screen.
Name		The name parameter of the object. This parameter must be different for each object because the object can only be accessed by the name defined here.
OnDown		Software that will run when the object is pressed when you click here is written on the page to be opened.
OnPress		Here is the software that will run when the object is held down is written on the page that will open when you click.

AIRHMI LCD SCREEN EDITOR MANUAL

OnUp		The software that will run when the object is pressed and released is written on the page that will open when you click here.
-------------	--	---

Functions

ShapeSet(" Object_name " , " Property to change " , " New_value ");

Description

Function that updates the parameters of the TransShape object.

void **ShapeSet**(unsigned char *name , unsigned char *type , unsigned char *value)

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Active setting command

ShapeSet (Object name , "Active" , "1 , 0 or True , False");

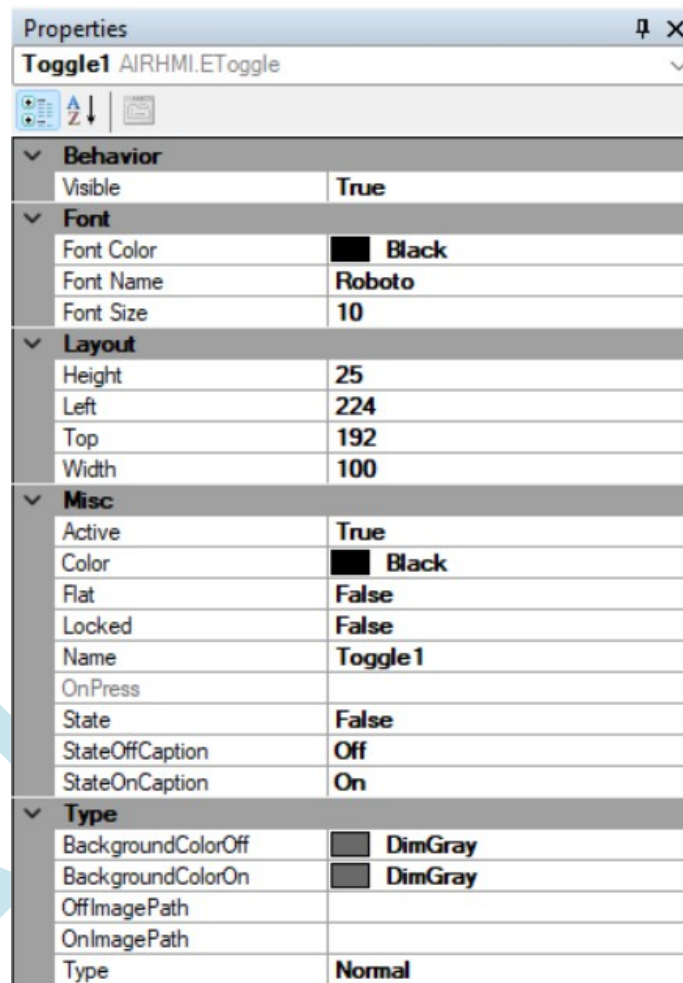
Sample Code:

ShapeSet ("Eshape1" , "Active" , "True");

6.11 Toggle

Toggle button is an AIR HMI user interface element used to change two different states (e.g. on/off, active/passive, present/absent) in an interface. The user can switch between these two states by clicking the toggle button. Toggle buttons are widely used, especially for operations such as enabling or disabling settings.

Toggle Properties Window



Feature	Option	Description
Visible	True	The screen appears the first time it is initialized.
	False	The screen is not visible when it is first initialized.
Font Color		Color option for the text on the Toggle object.
Font Name		Font option for the text on the Toggle object.
Font Size		Size option for the text on the Toggle object.
Height		Height parameter for the entire object (in pixels).

AIRHMI LCD SCREEN EDITOR

MANUAL		
Left		The position of the object on the horizontal axis parameter (in pixels).
Top		Parameter expressing the position of the object on the vertical axis (in pixels).
With		Width parameter for the entire object (in pixels).
Active	True False	Allows pressing the Toggle object. Does not allow pressing the Toggle object.
Color		Color parameter of the object (expressed in Hex is for example "#ffffff") .
Locked	True False	After moving the object to the desired location on the screen, if the property is <u>True</u> , it will stay there. If the property is <u>False</u> , you can set it to the desired position. It will make your work much easier when you work with multiple objects on the screen.
Name		The name parameter of the object. This parameter must be different for each object, because only the name defined here can be used to refer to the object can be reached.
OnPress		Software to run when the object is held down is written on the page that will open when you click here.
State	True False	Parameter that specifies in which state the toggle object will start (ON or OFF) when the screen is first created.
StateOffCaption		On the Toggle object when it is in the OFF state text parameter to write.
StateOnCaption		Text parameter to write on the Toggle object when it is ON.
BackgroundColorOff		Background when the Toggle object is OFF color (in Hex format, for example: "#ffffff").
BackgroundColorOn		Background color (in Hex format, for example: "#0000ff") when the Toggle object is OFF.
OffImagePath		When you want to make the Toggle object with your own images, you can choose the image from the field here. you can load it. This image will be active when the object is OFF.

AIRHMI LCD SCREEN EDITOR

OnImagePath	MANUAL	When you want to make the Toggle object with your own images, you can upload the image from here. This image will be active when the object is ON.
Type	Normal Image	The Toggle object will be selected when used in its default form. The Toggle object will be selected when used with your own images.

Functions

ToggleSet(" Object_name " , " Property to change " , " New_value ");

Description

Function that updates the parameters of the Toggle object.

void ToggleSet(unsigned char *name , unsigned char *type , unsigned char *value)

Parameter	Description
name	Name of the object
type	Name of the parameter of the object to be modified
value	The new value of the parameter to be modified

Visible adjustment command

ToggleSet(Object name , "Visible" , "1 , 0 or True , False");

Sample Code:

ToggleSet("Toggle1" , "Visible" , "True");

AIRHMI LCD SCREEN EDITOR

```
ToggleSet("Toggle1", "Visible", "False");
```

```
ToggleSet("Toggle1", "Visible", "1");
```

```
ToggleSet("Toggle1", "Visible", "0");
```

AIRHMI LCD SCREEN EDITOR MANUAL

Active setting command

ToggleSet (Object name , "Active" , "1 , 0 or True , False");

Sample Code:

```
ButtonSet("Toggle1" , "Active" , "True");
```

Font color adjustment command

ToggleSet(Object name , "Font_Color" , "Color code in Hex format for example: #0000ff"

); Sample Code:

```
ToggleSet("Toggle1" , "Font_Color" , "#0000ff");
```

Font size adjustment command

ToggleSet(Object name , "Font_Size" , "Font size is set between 8-102."); Sample

Code:

```
ToggleSet("Toggle1" , "Font_Size" , "12");
```

Height adjustment command

ToggleSet(Object name , "Height" , "Size (from 0 to Screen Y size)");

Sample Code:

```
ToggleSet("Toggle1" , "Height" , "25");
```

Width adjustment command

ToggleSet(Object name , "Width" , "Size (from 0 to Screen X size)"); Sample

Code:

```
ToggleSet("Toggle1" , "Width" , "100");
```

Ball adjustment command

ToggleSet (Object name , "Top" , "Y coordinate");

AIRHMI LCD SCREEN EDITOR MANUAL

Sample Code:

```
ToggleSet ("Toggle1" , "Top" , "255");
```

Left adjustment command

```
ToggleSet ( Object name , "Left" , "X coordinate" );
```

Sample Code:

```
ToggleSet ("Toggle1" , "Left" , "10");
```

Color adjustment command

```
ToggleSet ( Object name , "Color" , "RGB Color in hex format #RRGGBB" );
```

Sample Code:

```
ToggleSet ("Toggle1" , "Color" , "#FFA07A");
```

State setting command

```
ToggleSet ( Object name , "State" , "1 , 0 or True , False" );
```

Sample Code:

```
ToggleSet ("Toggle1" , "State" , "True");
```

```
ToggleGet(" Object_name " , " Property_to_retrieve " , " Variable_to_set_value_to ");
```

Description

Function that accesses and retrieves parameters of the Toggle object.

```
void ToggleGet(unsigned char *name , unsigned char *type , unsigned char *value)
```

Parameter	Description
name	Name of the object

AIRHMI LCD SCREEN EDITOR

type	MANUAL Name of the parameter of the object to fetch
Value	Variable to assign the parameter to be fetched

State Retrieval Command

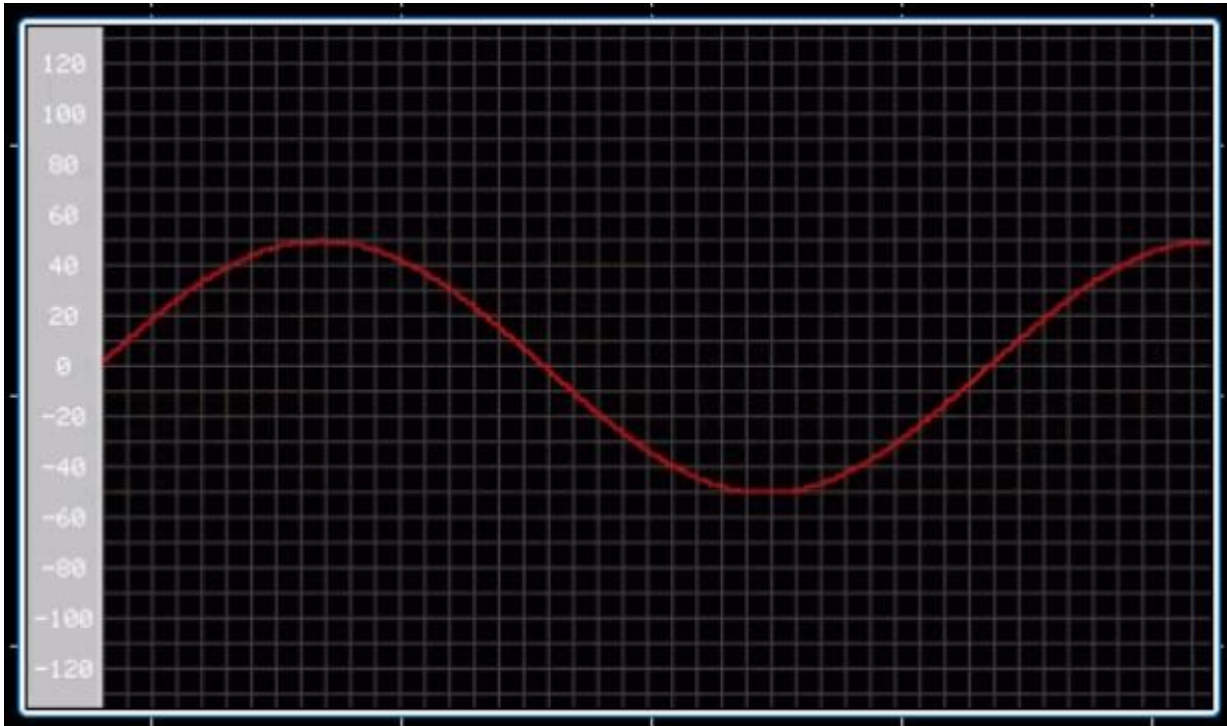
ToggleGet (Object name , "State" , "variable to assign");

Example code:

```
int get_state;  
ToggleGet ("Toggle1" , "State" , get_state);
```

6.12 Graph

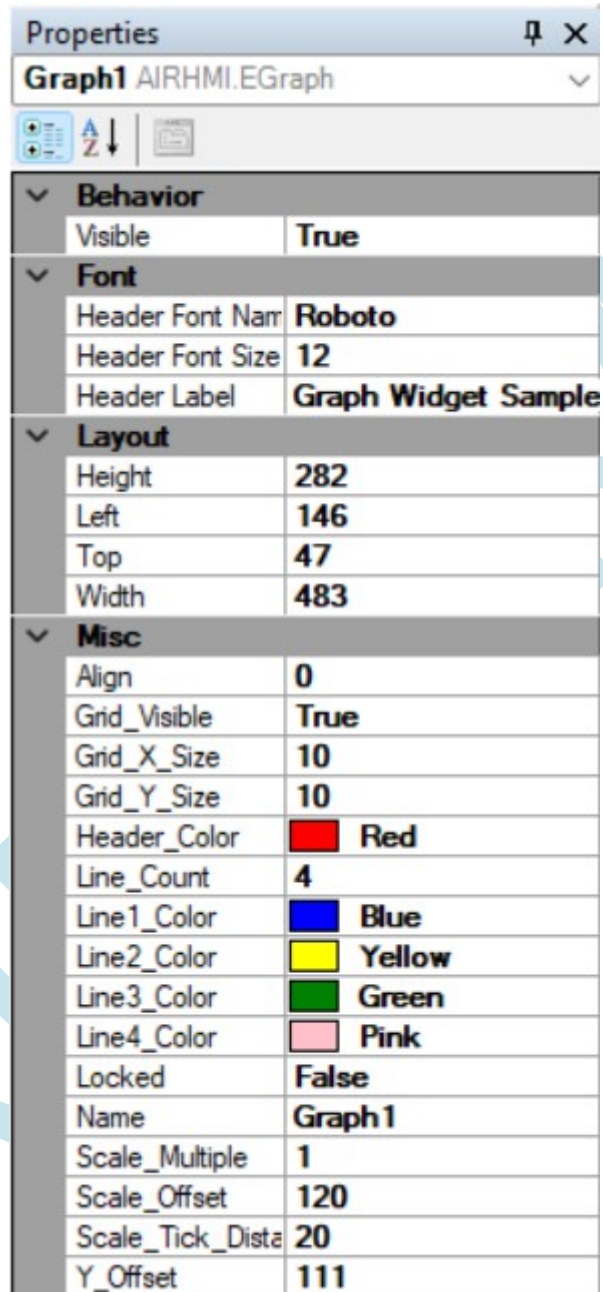
Used for drawing graphs on the screen. With this object you can make sense of and visualize your data.



AIRHMI

AIRHMI LCD SCREEN EDITOR MANUAL

Graph Properties Window



Feature	Option	Description
Visible	True	Appears when the screen first appears.
	False	The screen is not visible when it first appears.
Header Font Name		Defining different font options for Graph object It is done.
Header Font Size		The size of the font of the object's text.

A I R H M I LCD SCREEN EDITOR

Header Label	MANUAL	The text written in the title of the Graph object.
Name		The name of the object used for design. In the code section this name is used in the object name section.
Height		It is the height of the object.
Left		Specifies the position on the screen. X coordinate
Top		Indicates the position on the screen. Y coordinate
Width		It is the width of the object.
Align		Parameter used to align the Graph object.
Grid_Visible	True	Grids appear.
	False	Grids are not visible
Grid_X_Size		Pixel size of the cells in the grid on the X axis
Grid_Y_Size		Pixel size of the cells in the grid on the Y axis
Header_Color		Parameter for the color of the header.
Line_Count		Parameter to set how many lines on the graph object.
Line1_Color		Color parameter for the first line.
Line2_Color		Color parameter of the second line.
Line3_Color		Color parameter for the third line.
Line4_Color		Color parameter for the fourth line.
Locked	True	Does not allow to change the position of the object placed on the screen.
	False	You can move the chart to the desired position.
Scale_Multiple		Used to set the cell step amount on the Y axis.
Scale_Offset		Sets the offset value of the center of the graph on the Y axis.
Scale_Tick_Distance		Determines how often the values on the Y axis are written.
Y_Offset		It is the offset value of all data on the Y axis.

Functions

GRAPH_AddValue();

Description

Function used to add data to the chart object.

void **GRAPH_AddValue** (unsigned char *name , int channel, int value)

A I R H M I LCD SCREEN EDITOR

Parameter	Description
name	Name of the object
channel	How many line parameters of the object you want to update.
value	New value to be added.

GRAPH_AddValue(Object name , How many channels ,

value); Sample Code:

```
int data=10;
```

```
GRAPH_AddValue ("Graph1" , 1 , data); // continuously adds the value 10 to the graph.
```

Sample Code:

```
int data;
```

```
VarGet("EVariable1",&data);
```

```
GRAPH_AddValue ("Graph1" , 1 , data); // adds the data from the variable to the graph.
```

AIRHMI LCD SCREEN EDITOR MANUAL

GRAPH_Clear()

void **GRAPH_Clear** (unsigned char *name)

Description

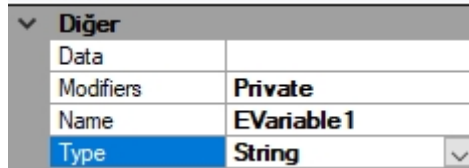
Function used to clear the chart object. Clears all drawn lines on the chart.

Parameter	Description
name	Name of the object

GRAPH_Clear (Object name);

GRAPH_Clear ("*Graph1*");

6.13 Variable



Diğer	
Data	
Modifiers	Private
Name	EVariable1
Type	String

Variables play a very important role in the code structure for situations where it is desired that the last values of the variables or the value of the variables in each edit in the code are not lost. Since the code structure is generally compiled and re-run every time the timer is active or when the touch is active in resistive screen projects, the normal variables created in it reset themselves. This poses major problems for the user when data from the previous position or state is to be used. To prevent such a problem, variables come into play. The name of the variables is given from the Attributes section with the Name heading. The type of the variable to be used should be selected under the Type heading as String if it is char and Integer if it is a numeric value. Another feature, Modifiers, should be selected from the Attributes section to select whether the variable we want to use will be Private (local) or Public (global). Local-global distinction is made in projects where more than one screen design will be used. If work will be performed on a single screen, the Private (local) variable can realize the desired state. However, in projects where it is desired to use more than one screen, for example, if a value on the second screen is desired to be used when switching to the first screen, the Public (Global) variable should be used here. Explanations on the use of variables in the code structure are given below.

VariableSave()

Description

AIRHMI LCD SCREEN EDITOR MANUAL

Saves the variable in the memory inside the screen. In this way, even if the screen is turned off and on, this variable value is kept in memory permanently. After making changes in the variable content, the same function is called again to save it again. Maximum 256 variables can be saved in memory.

Function

```
void VariableSave(unsigned char *name )
```

Parameter	Description
name	variable name

Example code

```
#include "stdio.h"
```

```
#include "stk.h"
```

```
VariableSave("EVariable1"); //
```

VarGet ()

Description

It is a data read command. Reads Private and Public variable. If

the variable is string, the direct name of the variable is written.

For example: char dataStr[20]; VarGet("var1",dataStr);

AIRHMI LCD SCREEN EDITOR MANUAL

If the variable is integer or double, the & sign is added in front of the name of the variable.

For example: `int dataInt; VarGet("var2",&dataInt);`

Function

`void VarGet(unsigned char *name , void *value)`

Parameter	Description
name	variable name
value	Pointer value according to variable type

Example code

```
#include "stdio.h"
```

```
#include "stk.h"
```

```
char dataStr[200];
```

```
int dataInt;
```

```
double dataDouble;
```

```
VarGet("EVariable1",dataStr);
```

```
VarGet("EVariable2",&dataInt);
```

```
VarGet("EVariable3",&dataDouble);
```

*If we want to read the content of the variable by Uart, we give NULL to the value part of the VarGet function, in this case it gives the content of the variable from the serial port.

```
VarGet("EVariable3",NULL);
```

AIRHMI LCD SCREEN EDITOR MANUAL

VarSet ()

Description

It is a data writing command. It can write Private and Public variable. If the variable is string, the direct name of the variable is written.

For example: `char dataStr[20]; VarSet("var1",dataStr);`

If the variable is integer or double, the & sign is added in front of the name of the variable.

For example: `int dataInt; VarSet("var2",&dataInt);`

Function

`void VarSet(unsigned char *name , void *value)`

Parameter	Description
name	variable name
value	Pointer value according to variable type

Example code

```
#include "stdio.h"
#include "stk.h"
char data[200];
int varint=5;
double varDouble = 2.15;
VarSet("EVariable1" , data);
VarSet("EVariable2" , &varint); // The value of EVariable2 becomes 5.
VarSet("EVariable3" , &varDouble); // The value of EVariable3 becomes 2.15.
```

AIRHMI LCD SCREEN EDITOR MANUAL

VarSet()

Function

void VarSet(unsigned char *name , int value)

Parameter	Description
name	Name of the local variable
value	Integer variable value

This function is used to assign an integer value directly to the variable. In the varSet function we need to pass an integer value as an address parameter, whereas in the varSet function we can give this value directly.

Example code

```
#include "stdio.h"

#include "stk.h"

VarSeti("EVariable1" , 15);

Int a = 5;

VarSeti("EVariable1" , a);
```


AIRHMI LCD SCREEN EDITOR MANUAL

VarSets()

Function

void VarSets(char *name , char *value)

Parameter	Description
name	Name of the local variable
value	String pointer variable

Example code

```
#include "stdio.h"
```

```
#include "stk.h"
```

```
VarSets("EVariable1" , "Hello World!"); Char
```

```
*data = "Hello World!";
```

```
VarSets("EVariable1" , data);
```

AIRHMI LCD SCREEN EDITOR MANUAL

VarSetf()

Function

void VarSetf(char *name , double value)

Parameter	Description
name	Name of the local variable
value	double variable value

Example code

```
#include "stdio.h"
```

```
#include "stk.h"
```

```
VarSetf("EVariable1" , 3.14);
```

```
double var = 3.14;
```

```
VarSets("EVariable1" , var);
```

StructGet ()

Description

In C, struct is used to organize multiple data types into a single structure. A struct can contain different types of variables and these variables are called struct elements. struct is similar to classes in object-oriented programming, but it does not contain functions.

Bir `struct` tanımlaması şu şekilde yapılır:

```
c Kodu kopyala  
  
#include <stdio.h>  
  
// Yapı tanımı  
struct Kisi {  
    char isim[50];  
    int yas;  
    float boy;  
};  
  
int main() {  
    // Yapıdan bir değişken oluşturma  
    struct Kisi kisi1;  
  
    // Değişkenlere değer atama  
    strcpy(kisi1.isim, "Ahmet");  
    kisi1.yas = 25;  
    kisi1.boy = 1.75;  
  
    // Yapı elemanlarına erişim ve ekrana yazdırma  
    printf("İsim: %s\n", kisi1.isim);  
    printf("Yas: %d\n", kisi1.yas);  
    printf("Boy: %.2f\n", kisi1.boy);  
  
    return 0;  
}
```

AIRHMI LCD SCREEN EDITOR MANUAL

Function

void StructGet(unsigned char *name , void *value)

Parameter	Description
name	variable name
value	pointer value according to struct type

Example code

```
#include "stdio.h"
#include "stk.h"

typedef struct
{
    int data1;
    int data2;
    char data3[100];
} data_t;

data_t data;
StructGet("data" , &data);
```

AIRHMI LCD SCREEN EDITOR MANUAL

StructSet ()

Description

Structe is used to read data.

Function

```
void VarSet(unsigned char *name , void *value , int size)
```

Parameter	Description
name	variable name
value	Pointer value according to variable type
Size	is the dimension of Structure.

Example code

```
#include "stk.h"  
#include "stdio.h"  
  
typedef struct  
{  
    int data1;  
    int data2;  
    char data3[100];  
} data_t;  
  
data_t data;  
  
data.data1 = 1;  
data.data2 = 2; sprintf(data.data3,"%s",  
"1234"); StructSet("data" , &data ,  
sizeof(data_t));
```

6.14 Delay()

Description

It is the command that waits for the specified time on the line it is used.

Function

void Delay (int ms)

Parameter	Description
ms	Specifies the time period

Example code

```
#include "stk.h"  
Delay(1000);
```

6.15 uartDataGet()

Description

According to the data coming from the UART, operations can be performed on the AMHI Editor screen. It is the command to receive data from the UART in the code layout.

Function

```
void uartDataGet(char *value , int *uartsize)
```

Parameter	Description
value	String to store the data from the UART
uartsize	Size of the data from the UART

Example code

```
#include "stdio.h"  
#include "stk.h"  
  
char uartData[3000];           // Store the data coming from the Uart  
string;                       // Size of the data coming from the Uart  
int uartsize;                 // Reading data from Uart  
  
uartDataGet(uartData , &uartsize);
```

**When sending data to the screen via Uart, you can communicate with the screen with functions such as VarSet, VarGet.*

You can watch the training video on the subject.



6.16 ChangeScreenSet ()

Description

It is the command that allows switching between the screens in the code.

Function

```
void ChangeScreenSet(unsigned char *value)
```

Parameter	Description
value	Name of the screen to switch to

Example code

```
#include "stk.h"
```

```
ChangeScreenSet("Screen1");
```


6.17 dateSet ()

Description

Command to refresh/set date data in the RTC.

Function

```
void dateSet ( unsigned char *days , unsigned char *months , unsigned char *years)
```

Parameter	Description
days	Day
months	Month
years	Year

Example code

```
#include "stdio.h"

#include "stk.h"

unsigned char day, month, year;           // Sample Date-Time variables in the code directory

day   = 10;
month = 2;
year  = 19;

dateSet(&day, &month , &year);         // Set date data from RTC
```

6.18 timeSet ()

Description

Command to refresh/set clock data in RTC.

Function

```
void timeSet(unsigned char *hours , unsigned char *mins )
```

Parameter	Description
hours	Clock
mins	Minutes

Example code

```
#include "stdio.h"  
#include "stk.h"  
  
unsigned char hour, min;           // Sample Date-Time variables in the code directory  
  
hour = 16;  
min = 30;  
  
timeSet(&hour , &min);           // Refresh/set clock data in RTC
```

6.19 dateGet ()

Description

Command to get date data from the RTC.

Function

void dateGet(unsigned char *days , unsigned char *months , unsigned char *years)

Parameter	Description
days	Day
months	Month
years	Year

Example code

```
#include "stdio.h"
#include "stk.h"
unsigned char day, month, year;           // Sample Date-Time variables in the code directory
dateGet(&day, &month , &year);         // Get date data from RTC
```

6.20 timeGet ()

Description

Command to receive clock data from the RTC.

Function

```
void timeGet(unsigned char *hours , unsigned char *mins )
```

Parameter	Description
hours	Clock
mins	Minute

Example code

```
#include "stdio.h"  
#include "stk.h"  
  
unsigned char hour, min;           // Sample Date-Time variables in the code directory  
timeSet(&hour , &min);          // Reading Clock data in RTC
```

6.21 AudioPlay()

Description

After the user adds the audio file they want to play to the project via AirHMI Editor, they can play it with this function.

Function

```
void AudioPlay(unsigned char *audioname , unsigned char volume)
```

Parameter	Description
audioname	Audio file name
volume	Sound level

Example code

```
#include "stdio.h"  
  
#include "stk.h"  
  
int volume; // Volume  
  
AudioPlay("AudioFileName" , volume );
```

6.22 AudioStop()

Description

Used to terminate the currently playing sound process.

Function

```
void AudioStop ();
```

Parameter	Description

Example code

```
#include "stdio.h"
```

```
#include "stk.h"
```

```
AudioStop();
```

6.23 AudioStatusGet()

Description

Sets whether the audio file is currently being played.

Function

```
void AudioStatusGet(int *value)
```

Parameter	Description
value	Player status (1 audio file is still playing, 0 audio file has finished playing).

Status query command

```
AudioStatusGet(int *value);
```

When the Value property is set to "True" the button object appears, when it is set to "False" it does not appear.

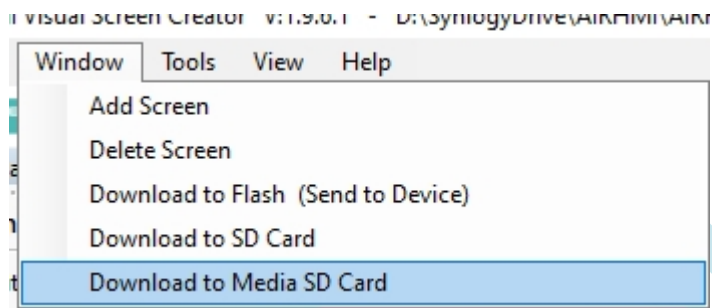
Sample Code:

```
int value;  
AudioStatusGet(&value);
```

6.24 VideoPlay()

Description

Used to play a video file. Plays the video in full screen. You need to transfer the video file to the sd card with the editor. For this, from the editor



You need to show your sd card by selecting the option. The format of the sd card must be FAT32.

Function

```
void VideoPlay(unsigned char *name , int volume);
```

Parameter	Description
Name	Video file name.
volume	Video volume.

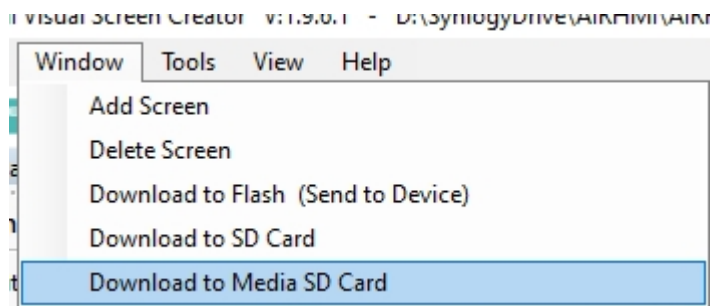
Example code

```
VideoPlay("EVideo1",100);
```


6.25 Video_Play_XY()

Description

Used to play a video file. Used to play the video at any position on the screen. You need to transfer the video file to the sd card with the editor. For this, from the editor



You need to show your sd card by selecting the option. The format of the sd card must be FAT32.

Function

```
void Video_Play_XY(char *name , int volume , int x , int y)
```

Parameter	Description
Name	Video file name.
volume	Video volume.
x	The screen is the x coordinate.
y	Screen y coordinate

AIRHMI LCD SCREEN EDITOR MANUAL

Example code

```
Video_Play_XY("EVideo1",100 , 10 , 20); // plays the video at coordinates 10,20 of the screen.
```

AIRHMI

6.26 File_write ()

Description

It is a command to write to Flash.

Function

```
void File_write(unsigned char *name , void *buffer ,int size , int nmemb)
```

Parameter	Description
name	Name of the .txt file to use
buffer	String array name
size	Size of the array to write
nmemb	1

Example code

```
#include "stdio.h"  
#include "stk.h"  
char x_file[200];  
memset(x_file , 0x00 , sizeof(x_file));  
sprintf(x_file , "%s" , "Hello World !!!");  
  
File_write("Message.txt" , x_file , sizeof(x_file), 1);  
  
// Created a file named Message.txt in Flash and put x_file in this file  
data written as sizeof(x_file).
```

6.27 File_read()

Description

It is a command to read from Flash.

Function

```
void File_read(unsigned char *name , void *buffer ,int size , int nmemb)
```

Parameter	Description
name	Name of the .txt file to use
buffer	String array name
size	Reading size
nmemb	1

Example code

```
#include "stdio.h"  
#include "stk.h"  
  
char x_file[200];  
memset(x_file , 0x00 , sizeof(x_file));  
  
File_write("Message.txt" , x_file , sizeof(x_file), 1);  
  
// From the data in a file named Message.txt in Flash  
Read sizeof(x_file) into variable x_file.
```

6.28 File_size()

Description

Command to learn file size.

Function

```
void File_size(unsigned char *name ,int *size)
```

Parameter	Description
name	Name of the file to use
size	An integer variable to hold the file size in

Example code

```
#include "stdio.h"  
#include "stk.h"  
  
int f_size;  
  
File_size("Message.txt" , &f_size); // Learn the size of the Message.txt file in  
Flash.
```

6.29 GPIO_Write()

Descriptio

n

Function

void GPIO_Write(unsigned char *portName ,int value)

Parameter	Description
portName	Gpio port
value	1 or 0

Example code

GPIO write command

```
GPIO_Write( GPIO name , 1 or 0 );
```

Sample Code:

```
GPIO_Write( "GPIO_1" , 1 );  
GPIO_Write( "GPIO_1" , 0 );
```

6.30 GPIO_Read()

Descriptio

n

Function

```
void GPIO_Read(unsigned char *portName ,int *value)
```

Parameter	Description
portName	Gpio port
value	1 or 0

Example code

GPIO read command

```
GPIO_Read( GPIO name , int *
```

```
); Sample Code:
```

```
int value;
```

```
GPIO_Read( "GPIO_1" , &value );
```

6.31 PWM_Set()

Description

There are 2 pwm outputs on the Airhmi display. With this function pwm frequency duty is adjusted.

Function

```
void PWM_Set(int ch , int freq , int duty);
```

Parameter	Description
ch	Pwm channel 1 or 0
freq	Pwm frequency
duty	Pwm 1 is the percentage of 0. Its value is given as 0-100.

Example code

PWM command

```
PWM_Set(int ch , int freq , int duty);
```

Sample Code:

```
PWM_Set( 0,1000000, 50 ); // Channel 0, 1Mhz 50% duty.  
PWM_Set( 1,2000000, 70 ); // Channel 0 , 2Mhz 70% duty.
```


6.32 BuzzerSet()

Description

The Airhmi display has a built-in buzzer.

Function

```
void BuzzerSet(int interval)
```

Parameter	Description
interval	Buzzer ring time in milliseconds.

Example code

Buzzer command

```
void BuzzerSet(int interval)
```

Example Code:

```
BuzzerSet( 100 ); // 100 ms buzzer is set.
```

6.33 I2C_Write()

Description

The Airhmi display has i2c capability.

Function

```
void I2C_Write(int speed , int deviceAddress , char *data , int dataLen)
```

Parameter	Description
speed	i2c communication speed
deviceAddress	i2c Slave device address
data	data
dataLen	Data length

Example code

I2C_Write command

```
void I2C_Write(int speed , int deviceAddress , char *data , int dataLen)
```

Sample Code:

```
Char data[] = {0xaa,0xbb,0xcc};
```

```
I2C_Write(10000, 0x55 , data , 3);
```

6.34 I2C_Read ()

Description

The Airhmi display has i2c capability.

Function

```
void I2C_Read(int speed , int deviceAddress , char *data , int dataLen)
```

Parameter	Description
speed	i2c communication speed
deviceAddress	i2c Slave device address
data	data
dataLen	Data length

Example code

I2C_Read command

```
void I2C_Read(int speed , int deviceAddress , char *data , int dataLen)
```

Sample Code:

```
Char data[3];
```

```
I2C_Read(10000, 0x55 , data , 3);
```

6.35 millis()

Description

The millis function allows the program to keep track of how long it takes to perform a certain function. For example, if it takes a certain amount of time to read data from a sensor and perform an action, this time can be tracked using the millis function. The millis function is simple to use. The function call returns the number of milliseconds elapsed since the start of the program. This value can be used by assigning it to a variable or used directly in a comparison expression.

Function

```
void millis(int *value)
```

Parameter	Description
value	It gives the elapsed time.

Example code

millis command

```
int startTime;  
millis(&beginningTime); // Saves the start time  
// Operations are  
performed int  
bitisZamani;  
millis(&bitisTime);  
if(endTime - startTime > 5000) {  
// 5 seconds passed  
}
```

6.36 KeypadAlpha()

Description

Used to retrieve data from the user during the software. A full page keyboard appears on the screen. The user enters data here using the keyboard and the keyboard return is assigned to a separate pointer.

Function

```
void KeypadAlpha(char *inData , char *outData)
```

Parameter	Description
inData	Text to be edited when the keyboard is opened,
outData	Keyboard return data.
data	data
dataLen	Data length

Example code

millis command

```
char data[100]; KeypadAlpha("Hello  
Dunya!",data);
```

```
printf("You Wrote %s.\n",data);
```

6.37 Modbus_ReadHoldingRegisters()

Description

In Modbus RTU protocol, function code "03" is used to read the holding registers of the device. This function code reads a subset of the holding registers starting from a specific starting address at a specified device address. The Modbus message used for this operation can be as follows:

For example, the following Modbus message can be used to read the 10 holding register of device 1 from address 4000:

Address: 01

Function Code: 03

Start Address: 4000 (0x0FA0)

Number of Holding Register to be read: 10 (0x000A)

After sending this message, the device's response is a Modbus message containing the values of the specified holding registers.

Function

```
void Modbus_ReadHoldingRegisters(unsigned char id, int address,int quantity, unsigned short * data, int timeout_ms);
```

Parameter	Description
id	Modbus id (0-255)
address	Modbus Slave Register Address
quantity	How many data to read

AIRHMI LCD SCREEN EDITOR

MANUAL

data	Modbus data
timeout_ms	Timeout value

Example code

```
#include "stk.h"  
#include "stdio.h"  
  
unsigned short data[2];  
  
Modbus_ReadHoldingRegisters(1,4000,2,data,1000);  
  
char resData[200];  
sprintf(resData,"%04x - %04x",data[0],data[1]);  
LabelSet("ELabel8", "Caption", resData );
```

6.38 Modbus_WriteSingleRegister()

Description

In Modbus protocol, function code "06" is used to write a single register value in a device. This function code writes a single data value to a specific register address at a specified device address.

The Modbus message used for function code "06" in Modbus RTU protocol is as follows:

Address: Device
address Function
Code: 06
Register Address: register address to write to
Value to Write: 16 bit data to write to the
register

For example, the following Modbus message can be used to write the value 1234 to address 4000 of device 1:

Address: 01
Function Code: 06
Register Address: 4000 (0x0FA0)
Value to Write: 1234 (0x04D2)

After sending this message, the device's response will not be a Modbus message. However, a confirmation message or error message can be received to make sure that the message has been sent.

Function

```
void Modbus_WriteSingleRegister(unsigned char id, int address ,unsigned short data, unsigned short *response, int timeout_ms);
```

Parameter	Description
id	Modbus id (0-255)
address	Modbus Slave Register Address

AIRHMI LCD SCREEN EDITOR

MANUAL

data	Modbus data
response	Modbus Slave device answer
timeout_ms	Timeout value

Example code

```
#include "stk.h"  
#include "stdio.h"  
  
unsigned short data[20];  
Modbus_WriteSingleRegister(1,4000,1234,data,1000);
```

6.39 Modbus_WriteMultipleRegisters()

Description

In Modbus protocol, function code "16" is used to write multiple register values. This function code writes multiple data values to a consecutive series of register addresses starting from a specific register address at a specified device address.

The Modbus message used for function code "16" in Modbus RTU protocol is as follows:

Address: Device
address Function
Code: 16
Start Address: record address to write to
Number of Records to Write: total number of records to write
Bytes to Write: the size in bytes of the number of data to write
Data: all register values to be written encoded in bytes sent consecutively

For example, starting from address 4000 of device 1, the 5 register values are respectively 1234,

The following Modbus message can be used to write 5678, 9101, 1121 and 3141:

Address: 01

Function Code 16

Start Address: 4000 (0x0FA0) Number

of Registers to Write: 5 (0x0005)

Number of Bytes to Write: 10 (0x0014)

Data: 04 D2 16 2E 23 29 04 49 0B 71

AIRHMI LCD SCREEN EDITOR MANUAL

Function

```
void Modbus_WriteMultipleRegisters(unsigned char id, int address , int quantity, unsigned short *data, unsigned char *response, int timeout_ms);
```

Parameter	Description
id	Modbus id (0-255)
address	Modbus Slave Register Address
qauantity	Number of data to be written to Modbus
data	Modbus data
response	Modbus Slave device answer
timeout_ms	Timeout value

Example code

```
#include "stk.h"  
#include "stdio.h"  
  
char data[20];  
unsigned short modbusData[2];  
  
modbusData[0] = 10;  
modbusData[1] = 11;  
Modbus_WriteMultipleRegisters(1,4000,2,modbusData,data,1000);
```

6.40 Modbus_ReadInputRegisters()

Description

In Modbus protocol, function code "04" is used to read input registers in a device. This function code reads a consecutive sequence of input registers starting from a specific input register address at a specified device address.

The Modbus message used for function code "04" in Modbus RTU protocol is as follows:

Address: Device address

Function Code: 04

Start Address: login registration address to be read

Number of Records to Read: total number of input records to read

For example, the following Modbus message can be used to read 5 input registers starting at address 10001 of device 1:

Address: 01

Function Code: 04

Start Address: 10001 (0x2711) Number
of Registers to Read: 5 (0x0005)

The device response will be a Modbus message containing the values of the requested input registers. The size of this message may vary depending on the number of input registers requested and the specific device features using the Modbus RTU protocol.

Function

```
void Modbus_ReadInputRegisters(unsigned char id, int address , int quantity, unsigned short *data, int timeout_ms);
```

Parameter	Description
id	Modbus id (0-255)
address	Modbus Slave Register Address

AIRHMI LCD SCREEN EDITOR

MANUAL

qauantity	Number of data to be written to Modbus
data	Modbus data
timeout_ms	Timeout value

Example code

```
#include "stk.h"  
#include "stdio.h"  
  
unsigned short data[20];  
Modbus_ReadInputRegisters(1,5000,2,data,1000);
```

7. Ethernet

7.1 Dhcp & Static ip identification

Description

DHCP (Dynamic Host Configuration Protocol) and static IP addresses are two different methods of assigning IP addresses used in computer networks. Here is how both methods work and their advantages:

DHCP (Dynamic Host Configuration Protocol):

DHCP is a protocol that automatically assigns IP addresses to devices on the network. Here is how it works:

When a device is connected to the network, network configuration information such as IP address, subnet mask, default gateway and DNS server address are automatically assigned to the device by the DHCP server.

The DHCP server manages IP addresses on the network and allows devices to dynamically obtain IP addresses.

This simplifies the management of IP addresses in large networks and allows devices to join the network automatically.

Static IP Addresses:

Static IP addresses are IP addresses that are manually assigned to each device and remain fixed without being changed. Here's how it works:

The network administrator or users assign an IP address, subnet mask, default gateway and DNS server addresses specifically for each device.

As these IP addresses are manually configured, each device has a fixed IP address and is not changed.

Function

```
void EthernetInit_DHCP();
```

A I R H M I LCD SCREEN EDITOR

Parameter	Description

Example code

```
#include "stk.h"
```

```
EthernetInit_DHCP();
```

Function

```
void EthernetInit_Static( char *ip , char *gw , char *netmask );
```

Parameter	Description
IP address	An IP address (Internet Protocol Address) is a unique identifier used for network communication between computers and other devices.
Gateway	Gateway, which enables data transmission between devices on a network is an important network device. Gateway facilitates data communication between two different networks or communication protocols.
Netmask	Netmask is a value used to distinguish between the network portion of an IP address and the portion of a device or subnet.

Example code

```
#include "stk.h"
```

```
EthernetInit_Static("192.168.1.150","192.168.1.1","255.255.255.0");
```

7.2 IP Address Inquiry

Description

An IP address (Internet Protocol Address) is an identifier used to communicate between computers and other network devices. IP addresses are used as part of the TCP/IP (Transmission Control Protocol/Internet Protocol) network protocol to ensure that devices can be found and communicate on the network. In networks such as the Internet and local area networks, each device has a unique IP address. IP addresses usually consist of four parts, and each part can have a value between 0 and 255. These four parts are written in dotted decimal format. For example, "192.168.1.1" is an example of an IPv4 (Internet Protocol version 4) IP address.

There are two basic types of IP addresses:

IPv4 (Internet Protocol version 4): This is the most commonly used type of IP address. IPv4 addresses are a 32-bit number and consist of 4 separate parts (each part has a value between 0 and 255). For example, "192.168.1.1" is an IPv4 IP address. However, as IPv4 addresses are running out, the transition to IPv6 has begun.

IP addresses ensure that devices are uniquely identified on the network and that data is routed correctly. In addition, IP addresses help the process of communicating on the network and enable important network functions such as access to the internet to take place.

Function

```
void EthernetGet_IP( char *ip_address);
```

Parameter	Description
ip_address	Airhmi is the ip address that the display receives from the server.

Example code

```
char data[100];
```

```
EthernetGet_IP(data);
```


7.3 MAC Address Inquiry

Description

The MAC address (Media Access Control Address) is a unique identifier that represents the hardware identification number of network devices. The MAC address is physically assigned on the network card or network interface and is usually 48 bits (6 bytes) long. The MAC address is used to identify devices during data transmission at the network level.

The MAC address is usually written in hexadecimal base and consists of six even digits. An example MAC address is as follows: "00:1A:2B:3C:4D:5E."

Function

```
void EthernetGet_MAC( char *mac_address);
```

Parameter	Description
mac_addres	The mac address of the Airhmi Ethernet interface.

Example code

```
char data[100];
```

```
EthernetGet_MAC(data);
```

7.4 Ethernet TCP Socket Connection

Description

Airhmi display can be used as static or DHCP.

Function

SocketTCP_Create("char * ip, int port);

Parameter	Description
IP address	An IP address (Internet Protocol Address) is a unique identifier used for network communication between computers and other devices.
Port Number	TCP Socket port number.

Example code

```
#include "stk.h"  
SocketTCP_Create("192.168.1.49",8000);
```

7.5 Ethernet TCP Socket Send Receive

Description

TCP socket is the function of sending and receiving data to and from the server.

Function

```
Void SocketTCP_SendReceive(char *sendData,char *rcvData);
```

Parameter	Description
sendData	The data we want to send to the TCP socket.
rcvData	TCP is the data received from the socket.

Example code

```
#include "stk.h"  
  
char DATA[1024];  
SocketTCP_SendReceive("GET {path} HTTP/1.1$0d$0aHost: {host}$0d$0a$0d$0a$0d$0a",DATA);  
printf("DATA:%s\n",DATA);
```

7.6 Send Ethernet TCP Socket

Description

TCP socket is the function of sending data to the server.

Function

```
Void SocketTCP_Send(char *SendData,int len);
```

Parameter	Description
sendData	The data we want to send to the TCP socket.
len	Data length

Example code

```
#include "stk.h"  
#include "stdio.h"
```

```
SocketTCP_Send("AIRHMI",6);
```

7.7 Ethernet TCP Socket AI

Description

TCP socket is the function of receiving data from the server.

Function

```
Void SocketTCP_Receive(char rcvData);
```

Parameter	Description
rcvData	Data received from a TCP socket.

Example code

```
#include "stk.h"  
#include "stdio.h"
```

```
char rcv[100];
```

```
SocketTCP_Receive(rcv);  
printf("Data:%s\n",rcv);
```

7.8 Ethernet TCP Socket Close

Description

TCP is a socket closure function.

Function

```
Void SocketTCP_Close();
```

Parameter	Description

Example code

```
#include "stk.h"  
#include "stdio.h"
```

```
SocketTCP_Close();
```

7.9 Ethernet TCP Socket Status Query

Description

Used for Airhmi TCP socket status query.

Function

```
Int SocketTCP_GetStatus();
```

Parameter	Description
10	Connected.
Others	Not connected

Example code

```
#include "stk.h"  
#include "stdio.h"  
  
int status = SocketTCP_GetStatus(); if(  
status == 10 )  
    LabelSet("ELabel3" , "Text" , "Connected." );  
else  
    LabelSet("ELabel3" , "Text" , " Not Connected." );
```

7.10 http post and get

Description

HTTP (Hypertext Transfer Protocol) is a protocol for communication between web browsers and web servers. HTTP provides two basic methods: GET and POST. These two methods are used for receiving, sending and processing web pages.

GET Method:

GET is used to make a request to the server to retrieve a specific resource (usually a web page).

A GET request is transmitted via URL and transfers data with query parameters added to the end of the URL.

A GET request does not send information to the server, it is only used to retrieve information.

If the GET request is reloaded in the browser or a link is clicked, the same request is repeated. Therefore, a GET request is idempotent, meaning that the result does not change when the same request is repeated.

The GET request appears in the browser history, so the data sent in the URL is clearly visible.

An example GET request with URL is as follows:

```
GET http://example.com/page.php?param1=value1&param2=value2
```

POST Method:

POST is used to send data to the server or update a resource.

A POST request adds the data to the body of the HTTP request, so it carries data in the body of the request, not through the URL.

Since POST request is used to transmit data, it is often preferred to securely send confidential information such as username, password and other sensitive information.

The POST request is not visible in the browser history, so the data sent is stored more securely.

AIRHMI LCD SCREEN EDITOR MANUAL

A POST request is not idempotent, meaning that the result may change when the

same request is repeated. An example POST request looks like this:

```
POST http://example.com/submit.php
```

Body:

```
param1=value1&param2=value2
```

Both HTTP methods serve different purposes in web applications. GET is usually used to retrieve information, while POST is used to send data and perform operations. The use of both methods depends on the needs of the application and security requirements.

8. Libraries

8.1 stdio.h

The "stdio" (Standard Input/Output) library is a widely used library in the C language. This library provides the tools needed for standard input/output functions. The functions in this library are used to input data from devices such as the keyboard and mouse, or to output data to the screen or to files. It is also used to handle and manage standard error and information messages.

```
int printf(char *, ...);
```

```
int fprintf(FILE *, char *, char
```

```
*, ...); int sprintf(char *, char
```

```
*, ...);
```

```
int snprintf(char *, int, char *, ...);
```

8.2 `stdlib.h`

The "stdlib" (Standard Library) library is a widely used library in the C and C++ programming languages. This library contains various functions and is mainly used for memory management, conversion operations, random number generation, program termination, file operations and other auxiliary functions.

The "stdlib" library is used in conjunction with the standard C library and is considered a standard library in the C programming language. It can also be used in C++.

Here are some of the functions included in this library:

Memory management functions (`malloc`, `calloc`, `realloc`, `free`)

Conversion functions (`atoi`, `atof`, `itoa`)

Random number generation function (`rand`)

Other helper functions (`abs`, `exit`, `qsort`)

These functions are common functions in the C language and are used in many programs. The "stdlib" library is a useful tool to improve code readability and speed up the development process.

```
float atof(char *);
```

```
float strtod(char *,char **);
```

```
int atoi(char *);
```

```
int atol(char *);
```

```
int strtol(char *,char **,int);
```

```
int strtoul(char *,char **,int);
```

```
void *malloc(int);
```

```
void *calloc(int,int);
```

```
void *realloc(void *,int);
```

```
void free(void *);
```

AIRHMI LCD SCREEN EDITOR MANUAL

`int rand();`

`void srand(int);`

`int abs(int);`

`int labs(int);`

AIRHMI

8.3 math.h

The "math" library is a widely used library in the C programming language. This library provides the tools needed for mathematical operations.

The functions in this library are used for trigonometric operations, exponential functions, logarithms, root calculations, range checking, number rounding and other mathematical operations.

Here are some of the functions available in the math library:

sin, cos, tan: Used for trigonometric operations. pow:

Used to calculate the exponent of a number. sqrt: Used

to calculate the square root of a number. exp: Used to

calculate the e^x of a number.

log, log10: Used to calculate the natural or decimal logarithm of a number. ceil,

floor: Used to round a number to the upper or lower integer.

fabs: Used to calculate the absolute value of a number.

These functions are used in many C programs that involve mathematical operations. math library is a useful tool to improve code readability and speed up the development process.

float acos(float);

float asin(float);

AIRHMI LCD SCREEN EDITOR MANUAL

float atan(float);

float atan2(float, float);

float ceil(float);

float cos(float);

float cosh(float);

float exp(float);

float fabs(float);

float floor(float);

float fmod(float, float);

float frexp(float, int *);

float ldexp(float, int);

float log(float);

float log10(float);

float modf(float, float *);

float pow(float, float);

float round(float);

AIRHMI LCD SCREEN EDITOR MANUAL

float sin(float);

float sinh(float);

float sqrt(float);

float tan(float);

float tanh(float);

AIRHMI

8.4 string.h

The "string" library is a widely used library in the C programming language. This library provides the tools needed for string operations.

The functions in this library are used for operations related to character strings. These operations include concatenation, comparison, copying, length calculation and other operations.

Here are some of the functions available in the string library:

`strcpy`: Used to copy a string of characters to another string of characters. `strcat`:

Used to concatenate two strings of characters.

`strlen`: Used to calculate the length of a string. `strcmp`: Used to compare two strings.

`strchr`: Used to search for a specific character in a character sequence.

`strstr`: Used to search for a specific substring in an array of characters.

These functions are frequently used in the C programming language for operations with strings. The string library is a useful tool to improve the readability of code and speed up the development process.

AIRHMI LCD SCREEN EDITOR MANUAL

Sample Program:

```
#include <stdio.h>
#include <string.h>

int main() {
    char string1[20] = "Merhaba";
    char string2[20] = "dünya";
    char string3[40];

    // string1 ve string2 karakter dizilerini birleştirir
    strcat(string1, string2);

    // Birleştirilmiş karakter dizisini string3'e kopyalar
    strcpy(string3, string1);

    printf("Birleştirilmiş karakter dizisi: %s\n", string1);
    printf("Kopyalanan karakter dizisi: %s\n", string3);

    // string1 karakter dizisinde "dünya" alt dizisi aranır
    if (strstr(string1, "dünya") != NULL) {
        printf("string1 karakter dizisinde 'dünya' bulundu.\n");
    } else {
        printf("string1 karakter dizisinde 'dünya' bulunamadı.\n");
    }

    // string1 ve string3 karakter dizileri karşılaştırılır
    if (strcmp(string1, string3) == 0) {
        printf("string1 ve string3 karakter dizileri eşittir.\n");
    } else {
        printf("string1 ve string3 karakter dizileri eşit değildir.\n");
    }

    return 0;
}
```

AIRHMI LCD SCREEN EDITOR MANUAL

Program Output:

```
Birleştirilmiş karakter dizisi: Merhabadünya
Kopyalanan karakter dizisi: Merhabadünya
string1 karakter dizisinde 'dünya' bulundu.
string1 ve string3 karakter dizileri eşittir.
```

```
void *memcpy(void *,void *,void
*,int); void *memmove(void *,void
*,void *,int); void *memchr(char
*,int,int);
int memcmp(void *,void
*,int); void *memset(void
*,int,int); char *strcat(char
*,char *); char *strncat(char
*,char *,int); char *strchr(char
*,int);
char *strrchr(char *,int);
int strcmp(char *,char *);
int strncmp(char *,char *,int);
int strcoll(char *,char *);
char *strcpy(char *,char *);
char *strncpy(char *,char *,int);
char *strerror(int);
int strlen(char *);
int strspn(char *,char *);
int strcspn(char *,char *);
char *strpbrk(char *,char *);
char *strstr(char *,char *);
char *strtok(char *,char *);
int strxfrm(char *,char *,int);
```