



## **Controllers Programming**

### **General**

Writing a program for controllers is done by using “WinBak” software under the windows platform. The program writing is done graphically in the programming screen (These instructions concerns UniPoint NT and UniSense controllers only).

The following instructions are divided into 2 main sections:

- I. Presenting the programming screen and its areas.
- II. A program writing demonstration step by step.

Beside these sections there are 2 appendixes:

Appendix 1 - Before Programming.

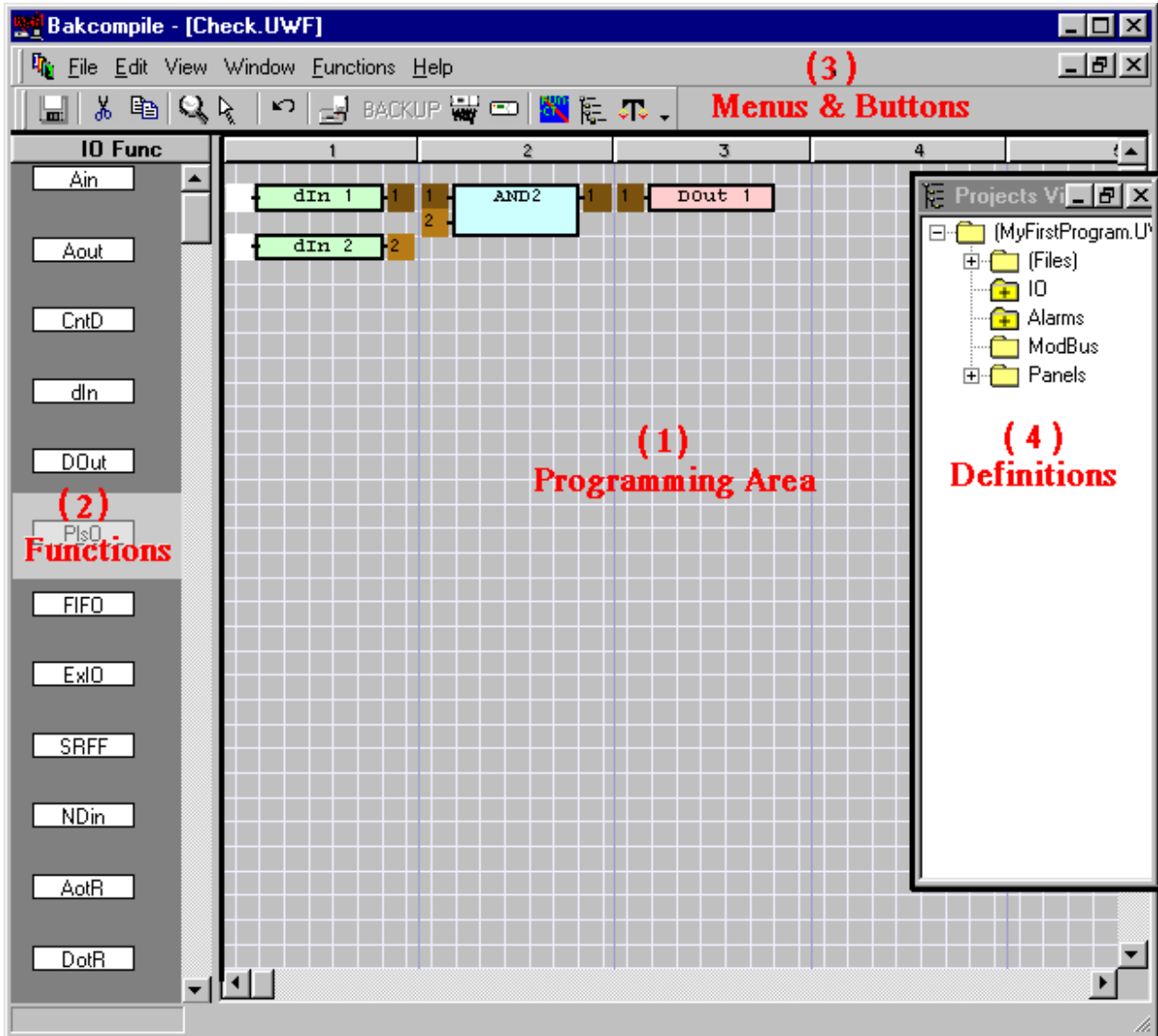
Appendix 2 - The Project View Tree.

[To go directly to the programming demonstration, press here.](#)



## I. The Programming Screen

This is the main controller's programming window:



Picture No. 1

This screen is used for writing programs and sending them to the controllers. The programming is done by dragging functions from the left area of the screen ([Area 2](#)) to the programming area ([Area 1](#)), and connecting these functions together.



### **The Programming Area (1)**

This area is divided into columns in which it is possible to place different programming functions from the functions pool located at the left part of the screen (2). Afterwards, the placed functions should be wired (connecting Input to Output). Wiring can take place **ONLY** between functions placed in following columns! For example: The Output of functions placed in the 1<sup>st</sup> column can be wired only to Inputs of functions placed in the 2<sup>nd</sup> column, the Output of functions placed in the 2<sup>nd</sup> column can be wired only to Inputs of functions placed in the 3<sup>rd</sup> column, etc'.

In [picture no. 1](#) we placed digital Inputs Din1/Din2 in column no. 1, AND gate in column no. 2, and digital Output Dout1 in column no. 3. After wiring, this program will operate Dout1 only if Din1 and Din2 are in position “1”.

Functions that are placed already can be dragged again by clicking on them with the mouse, and dragging after a short pause. In addition, clicking on a function with the right button of the mouse will open a menu with an additional action.

### **The Functions Area (2)**

This area contains the available programming functions. The functions are divided into categories. Selecting a category is done by clicking on the chosen category button, and then clicking on the desired function (the default in picture no. 1 is “IO Function”).

### **The Menus and Buttons Area (3)**

This area contains menus and buttons that enable different actions, such as save, edit, and print, as well as debugging and compiling, etc.

### **The Definitions Area (4)**

This area contains the program definitions: the program files, the IO tables, alarms, communication with sub units and panels, etc..


[For additional details see appendix 2.](#)



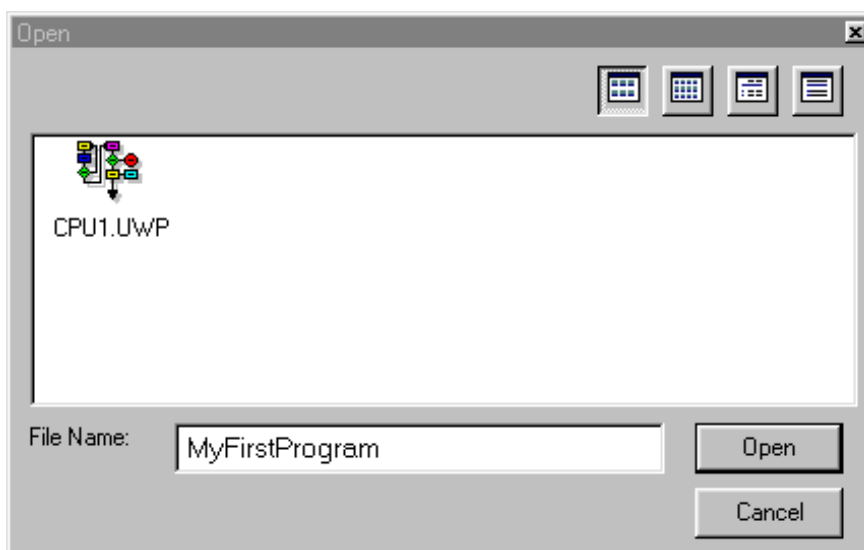
## II. My First Program

Following are the instructions for writing a simple program for a controller that simulates opening a gate by pressing two buttons simultaneously (“AND” gate).

1. In order to start programming you need to run WinBak software, by clicking on

the controller icon  (programming) in the left toolbar of the UniArt HMI software. Clicking on this button will open the entrance screen (open) of the WinBak.

[If you received a different screen - see appendix 1 for changing the settings.](#)



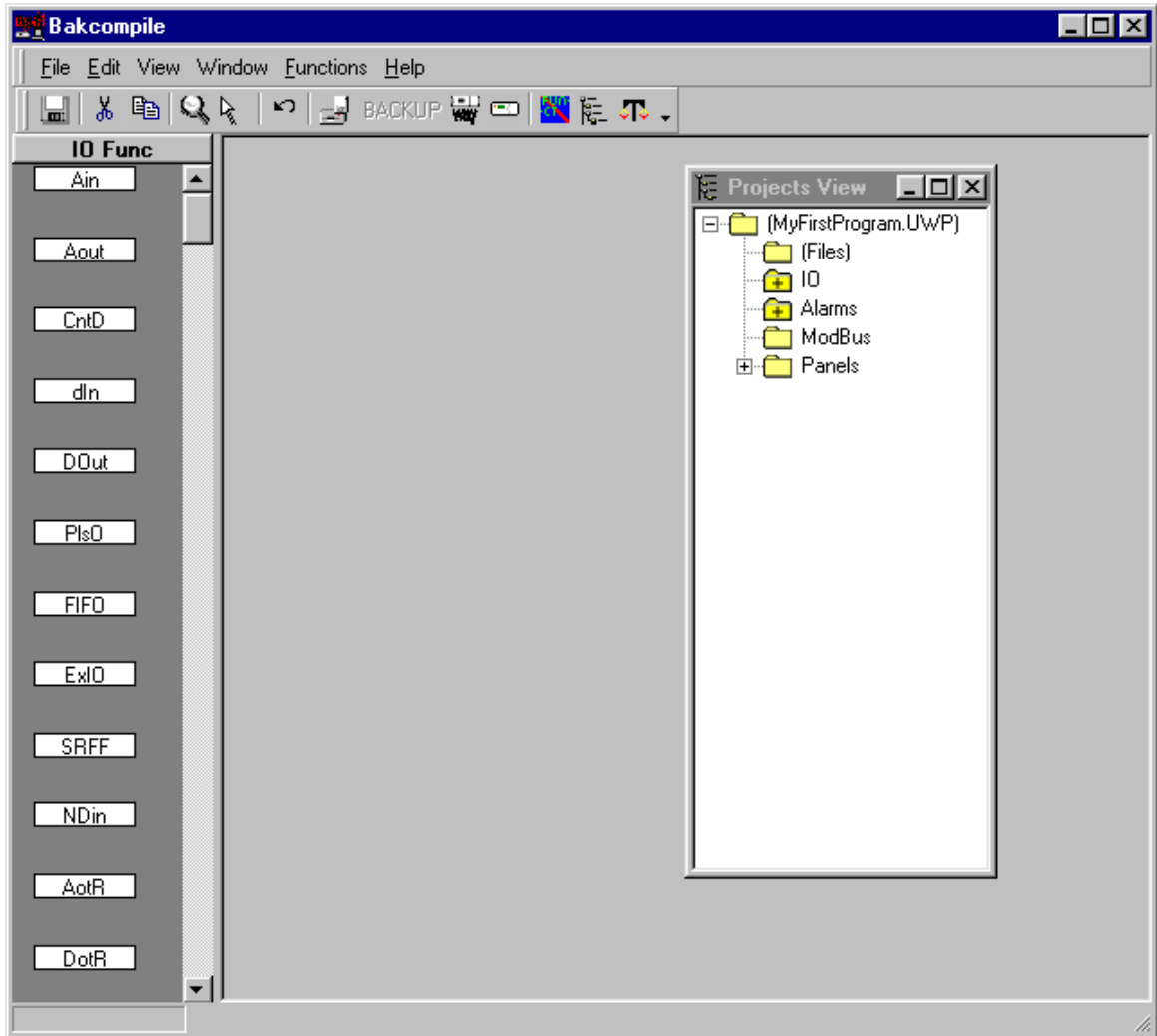
Picture No. 2

In this screen it is possible to choose an existing project by clicking on it, or to create a new one by typing a new name.

Type the name “MyFirstProgram” in order to create a new project for practice.



The following screen will open:

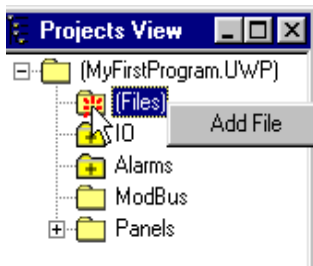


Picture No. 3



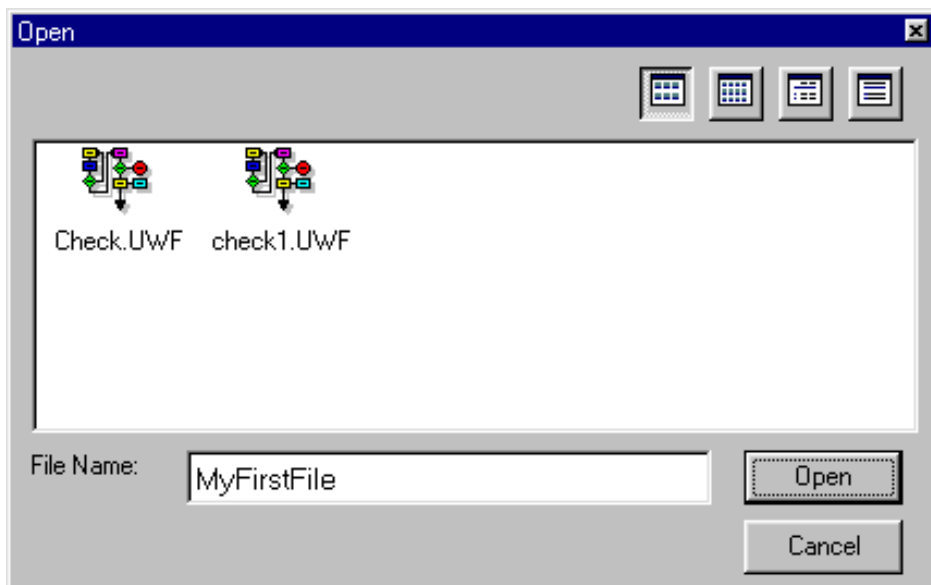
- The next step is creating a programming file. In order to do that you have to click with the right button of the mouse on the branch titled “files” (in picture no. 3) and choose “Add Files” from the menu that opens.

[For additional details see appendix 2.](#)



Picture No. 4

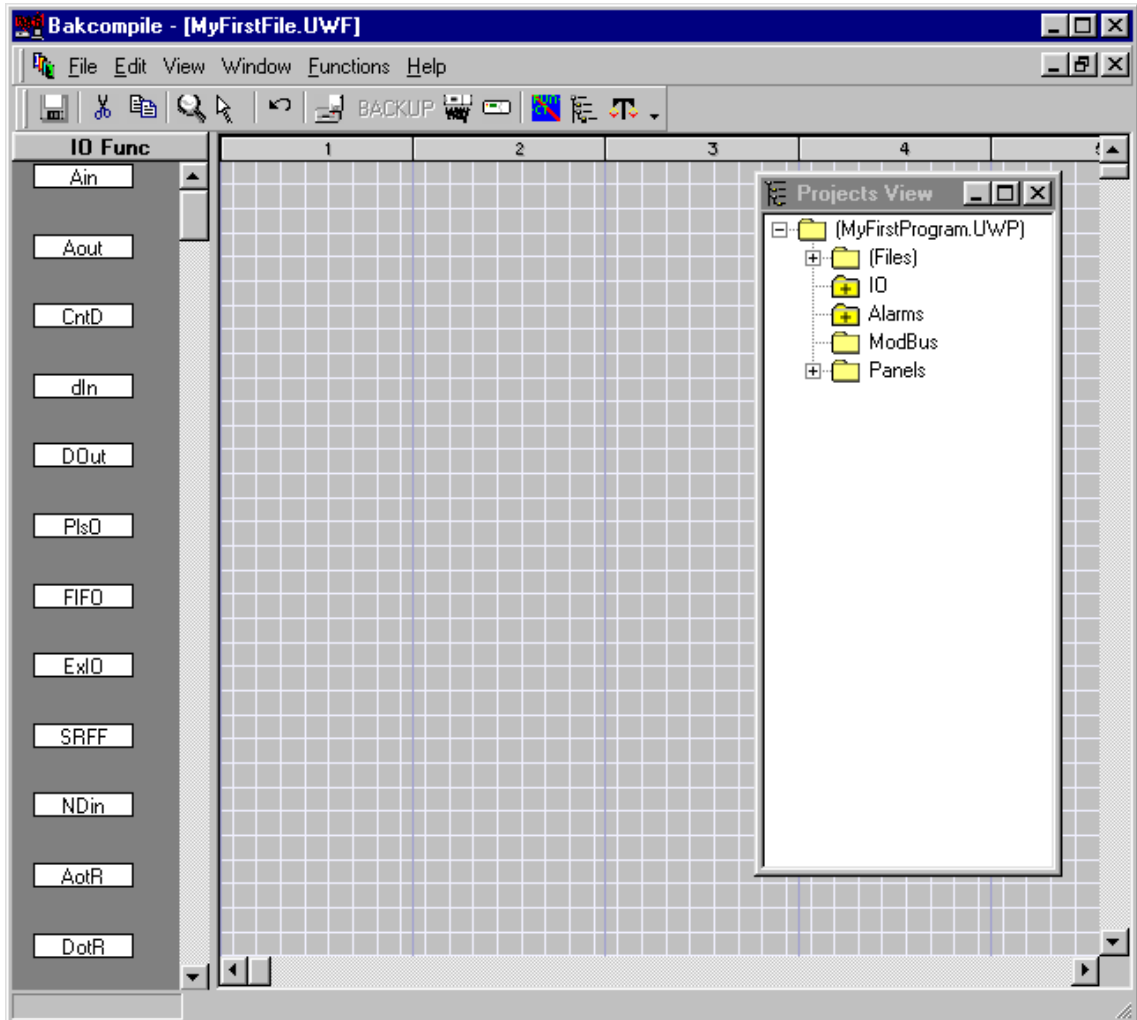
Here two, it is possible to choose an existing file or type a name for a new one. Type the name “MyFirstFile” in order to create a new file for practice.



Picture No. 5



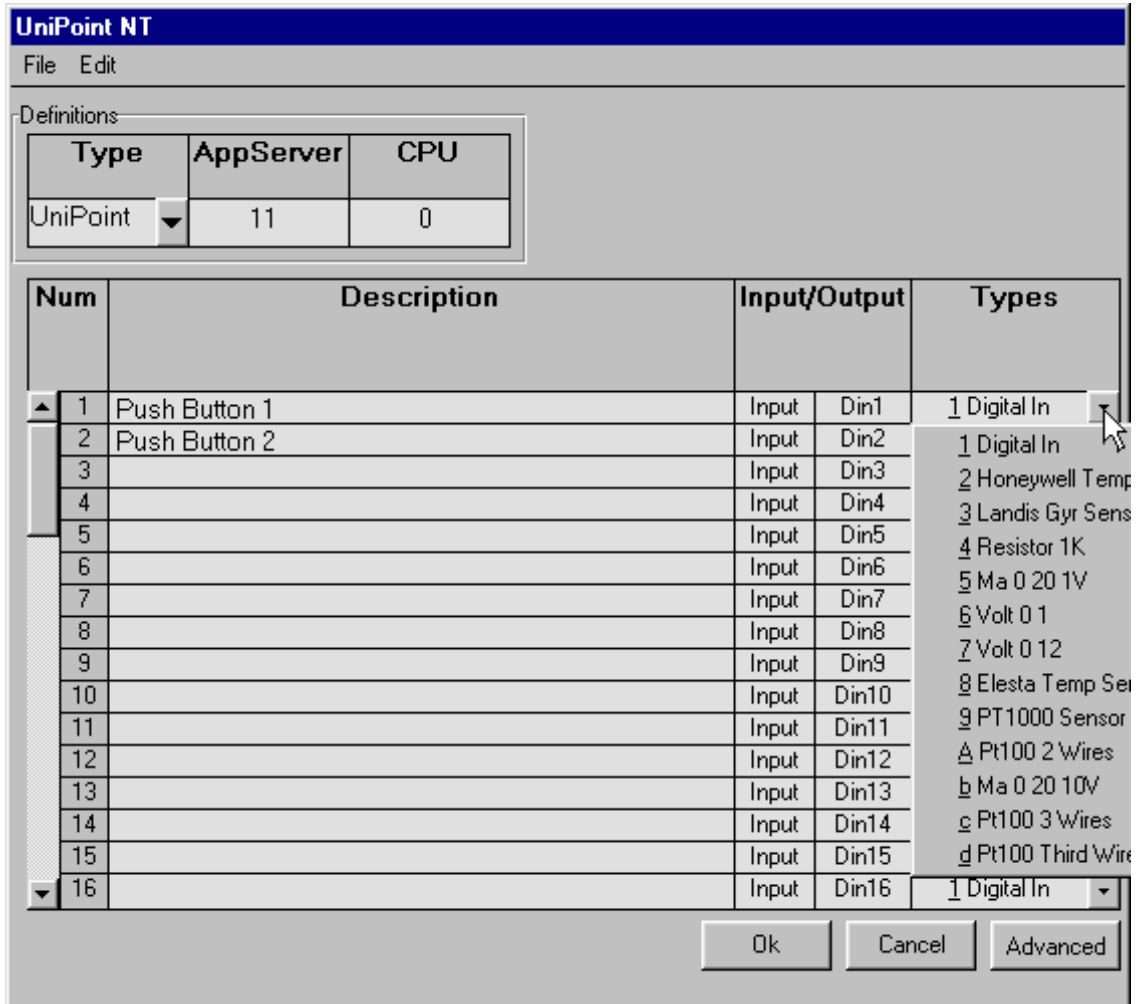
After opening a new file the following screen will appear:



Picture No. 6



- The next step is defining the Inputs and the Outputs. Double click on the branch titled “IO” (in picture no. 3) to open the following screen:



Picture No. 7

This screen is used to define a description (name) and a type for each IO point. In order to define the point’s type click on the “down arrow” on the right side of the type column, and choose the desired type. In the practice program we are writing we will use Digital in 1 for the first push button, and Digital in 2 for the second push button.

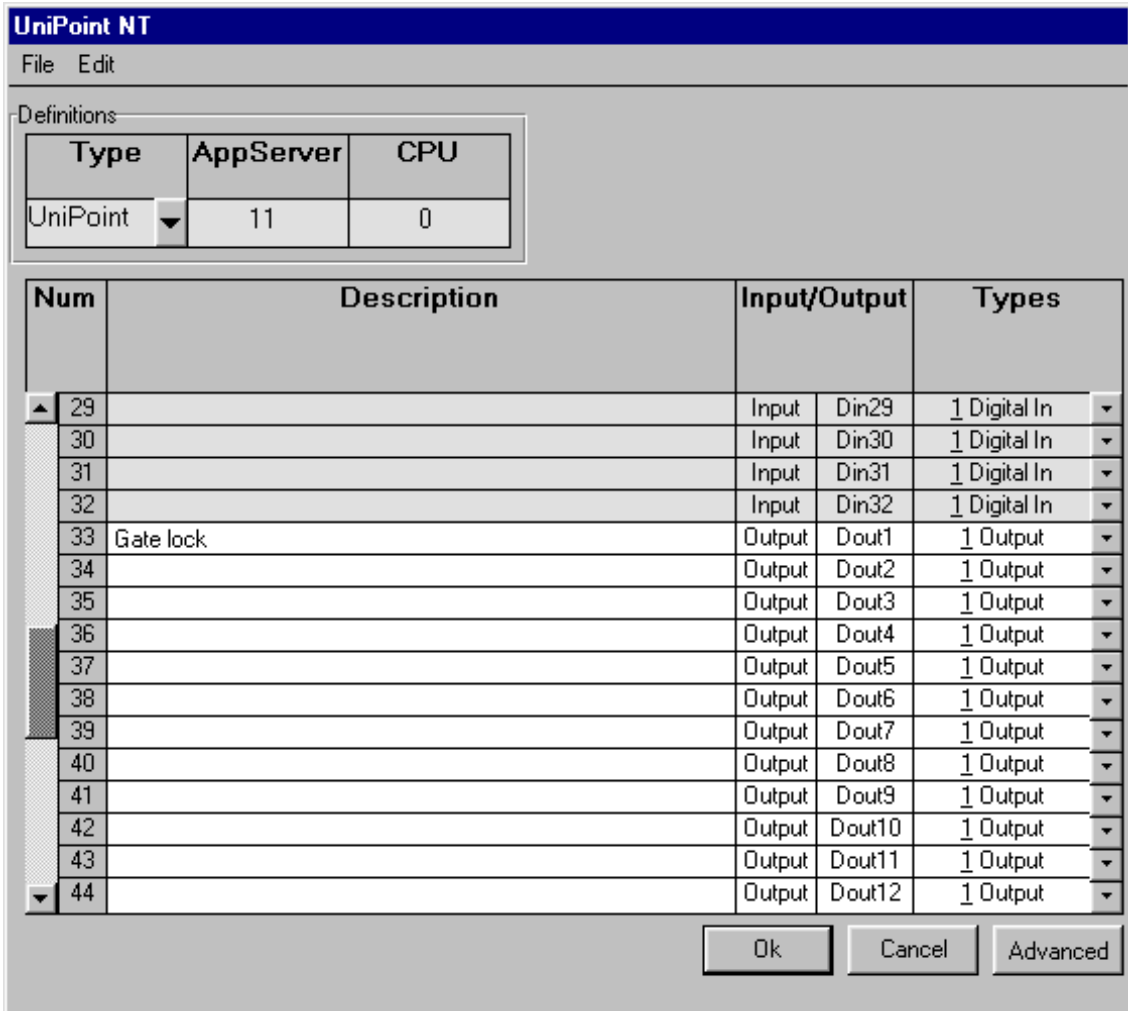
Type “Push Button 1” in line 1. In this case it is unnecessary to redefine the type, since the default is “Digital In”.

Type “Push Button 2” in line 2. In this case as well it is unnecessary to redefine the type, since the default is “Digital In”.





Roll the screen downwards in order to get to the Outputs definition area:



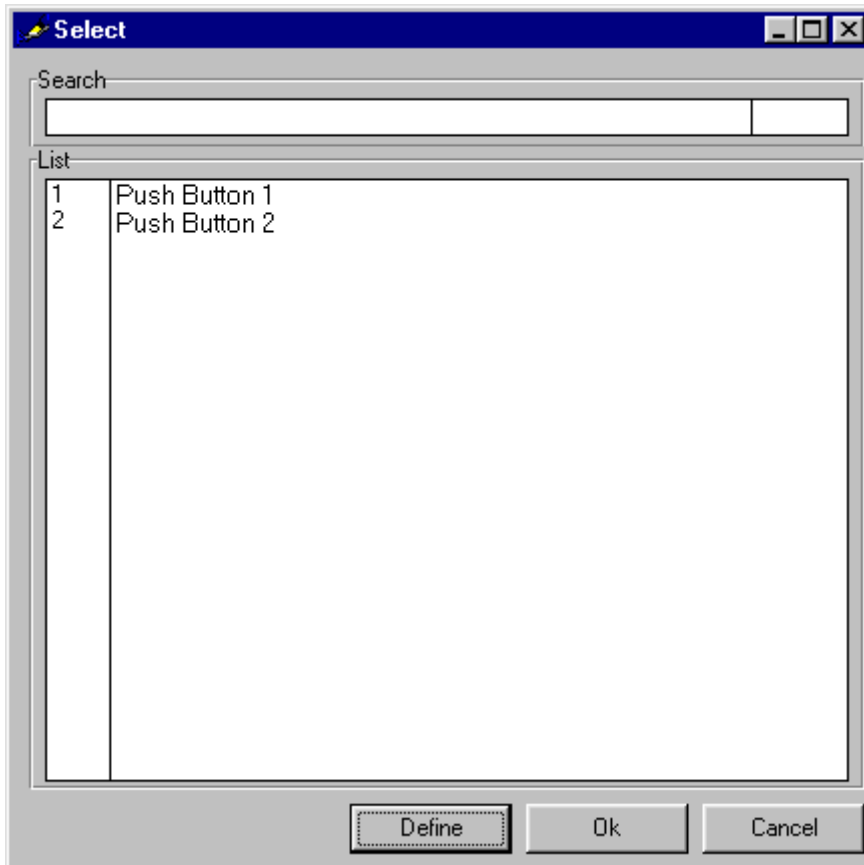
Picture No. 8

Type “Gate Lock 1” in row 33, which is connected to Output 1. In order to save your definitions and go back to the programming screen, click on the “OK” button on the screen or “ENTER” in the keyboard.



4. After completing the IO definitions, it is time to start the programming itself ([see picture no. 1](#)).

Drag a Digital In from the left area of the screen ([Area 2](#)) to column 1 in the programming area ([Area 1](#)). This action will open a screen with the Inputs we defined earlier:

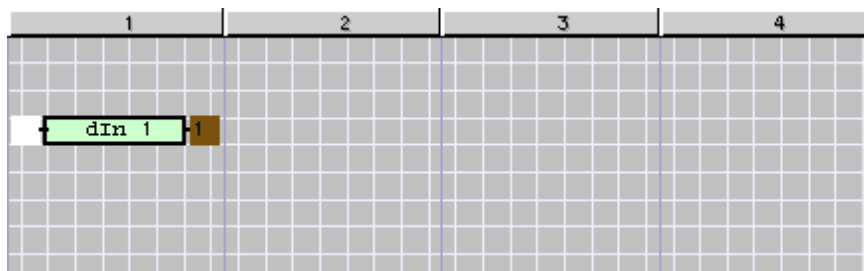


Picture No. 9

Choose the first Input and click on the “OK” button.

Choose the first Input and click on the “OK” button.

The first Input will then appear in the programming area like this:

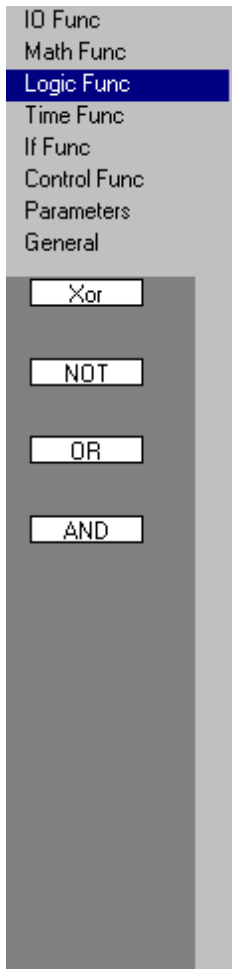


Picture No. 10

Repeat the procedure with the second Input.



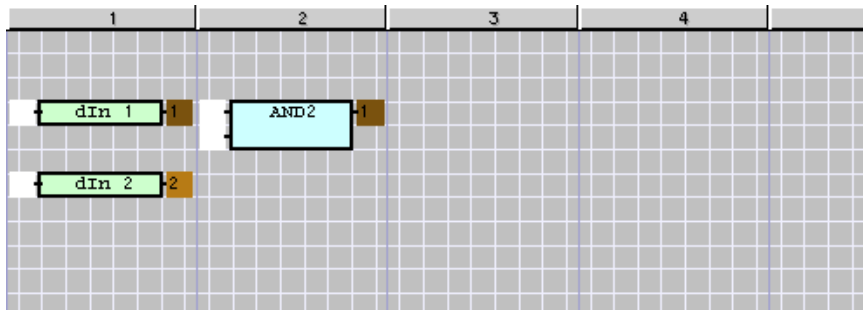
5. In order to place an “And” gate, choose “Logic Func” from the functions area on the left ([Area 2](#)).



Click on the “IO Function” in [Area 2](#), and choose “Logic Func”. Drag an “AND” function to the second column in the programming area ([Area 1](#)). In the menu that pops up choose “AND 2”.

Picture No. 11

Now, the programming area ([Area 1](#)) will look like this:



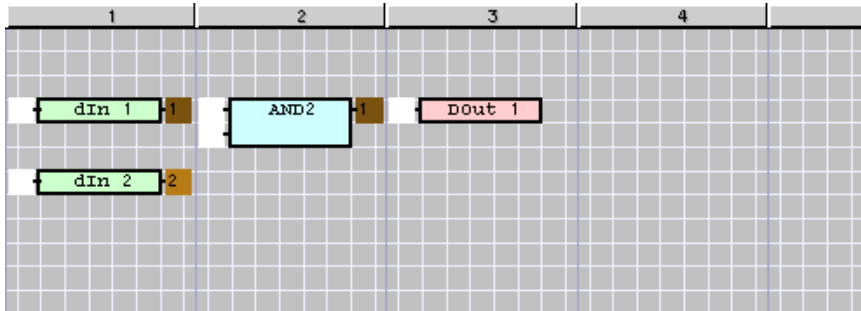
Picture No. 12



6. In order to place a “digital Output” choose the “IO Func” from the functions area on the left ([Area 2](#)).

Drag a digital Output to the third column, and from the menu that opens choose the digital Output “Gate Lock” we defined earlier.

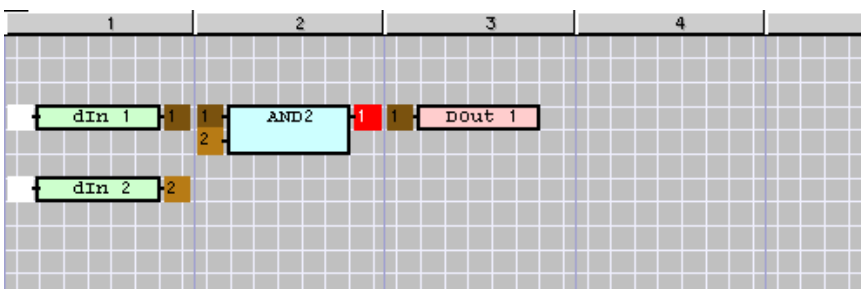
At the end of this stage the programming area ([Area 1](#)) will look like this:



Picture No. 13

7. Wiring: The process of connecting the different functions together is called wiring. In order to wire a program click on a functions’ Output, and then on the Input of the functions needed to be wired.
- A. Click on the Output of Din1. The number “1” in the Output will get a red background.
  - B. Click on the first (upper) Input of the “AND” gate (the Input is represented by the white area on the left side of each function). The number “1” with a brown background will appear instead of the white background.
  - C. Repeat this process with the Output of Din2 and the second (lower) Input of the “AND” gate.
  - D. Use the same technique to wire the Output of the “AND” gate to the Input of the digital Output in the third row.

The program is now complete, and the programming area ([Area 1](#)) will look like this:

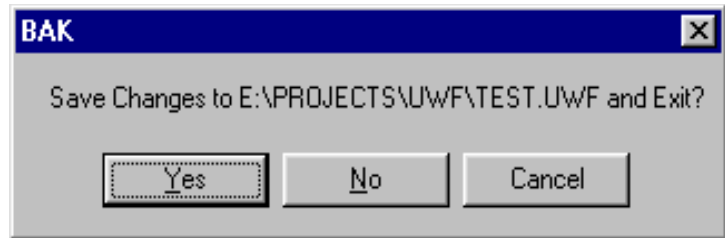


Picture No. 14



8. Saving: After the programming is done you need to save it. Saving is done by one of these three ways:

A. When exiting the programming area (when closing the window) you will be asked whether you want to save the changes.



Picture No. 15

Click “Yes” for saving, “No” for exiting without saving, and ”Cancel” to abort exiting.

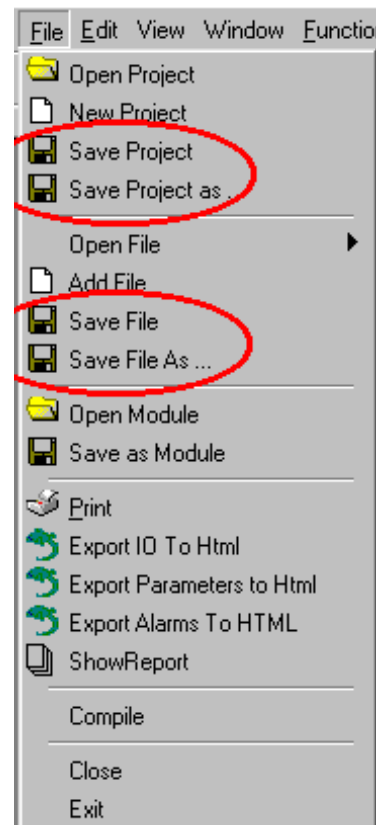
B. Click on the “Save” button in the toolbar located in the menus



Picture No. 16

and buttons area ([Area 3](#)).

C. Click on the “File” menu in the toolbar located in the menus and buttons area ([Area 3](#)), and choose the desired option.



Picture No. 17


9. Downloading: The last step is to download the program to the controller, but it will not be detailed in this summary.



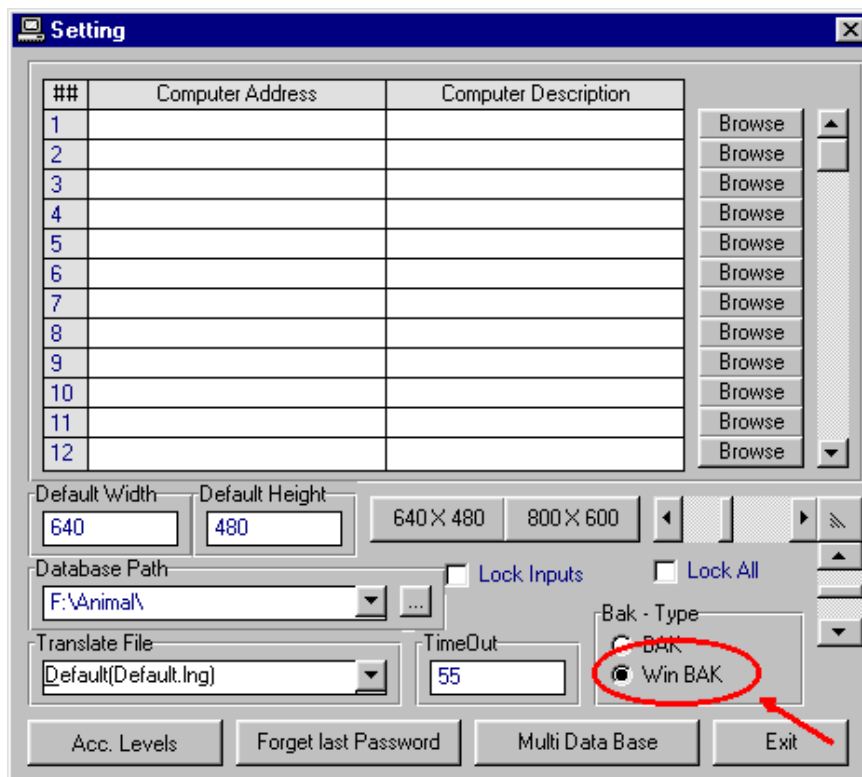
## Appendix 1 - Before Programming

Before starting the programming process, you need to define the working area.

In order to do this you need to enter the “Settings” screen of the main menu, by clicking

on the third button from the bottom in the left  toolbar.

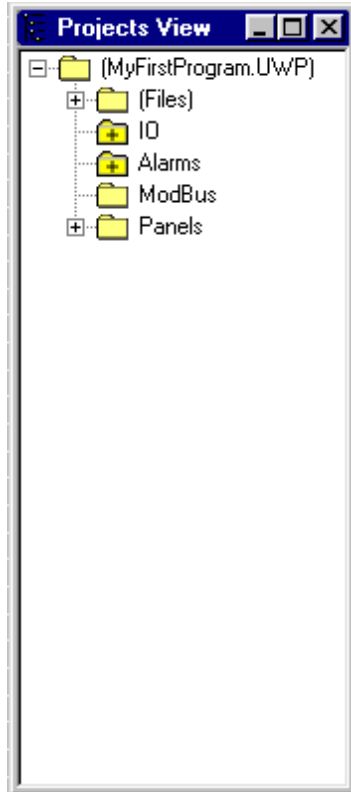
In the Settings window that opens choose “WinBak” for programming in windows area (for UniPoint NT controllers).



Picture No. 18





## Appendix 2 - The Project View Tree

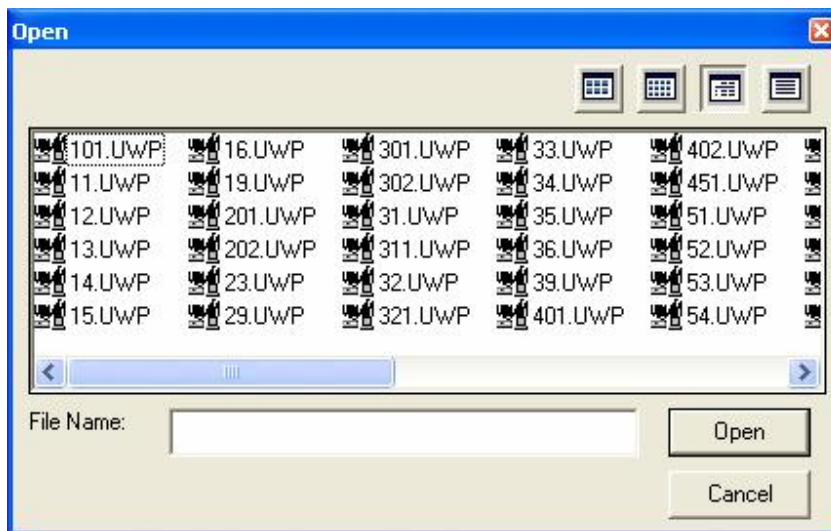


Picture No. 19

- ü Files - A list of the program's files (a program is composed of one file or more). In order to add files to this branch click with the right button of the mouse on the root of this branch (Files) and choose "Add Files". Double click on an existing file will present it in editing mode.
- ü IO - Definitions of the controller's Inputs and Outputs. Double click on it will open the IO Definitions screen.
- ü Alarms - A list of the the controller's alarms. An alarm in the UniArt-HMI can be translated as a pop up alarm or event. Double click will open the definitions screen. Will not be detailed in this summery.
- ü ModBus - The controller can be ModBus master. Will not be detailed in this summery.
- ü Panels - Definitions of the control panels connected to the controller. Will not be detailed in this summery.

## SuperBrain – WinBak programming – Beta Version

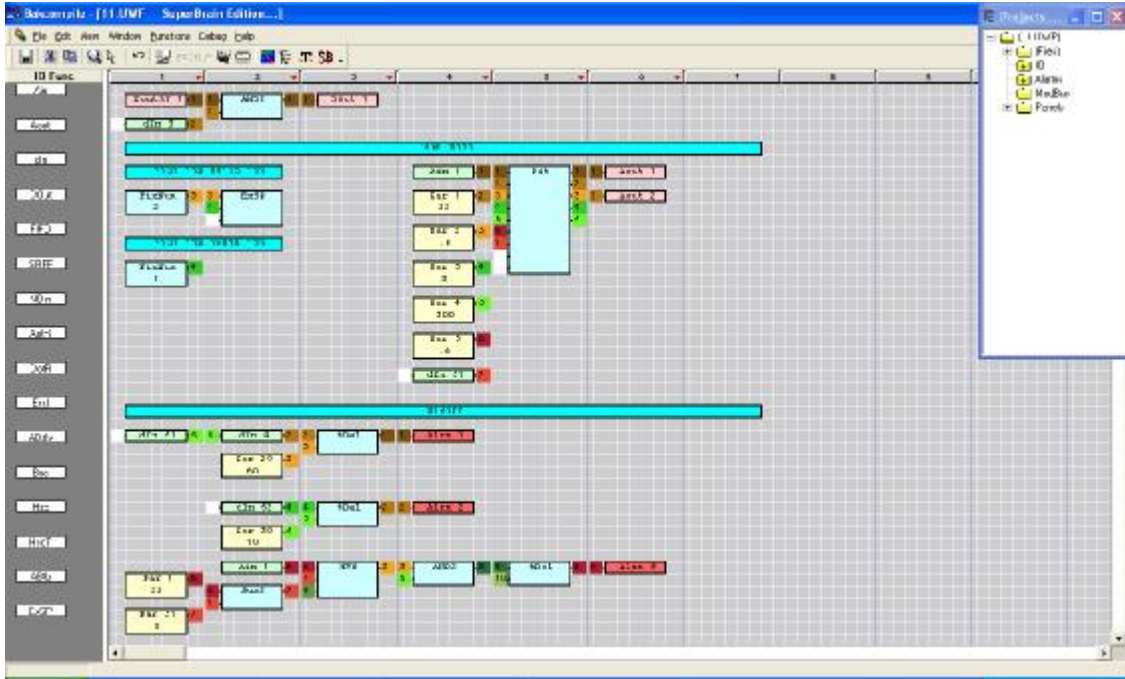
1. Installation over existing Uniart:
  - 1.1. copy files: *WinBAK\_SBEx1.exe* and *SbBakDef.uni* to *c:\Uniart\bin*.
2. Creating a project:
  - 2.1. Make a new project from Uniart Main-Menu  and set the project path: 
  - 2.2. Run *WinBAK\_SBEx1.exe*.
  - 2.3. Under the project folder the following folders will be created automatically:
    - 2.3.1. **UWP** – for Pprojects files.
    - 2.3.2. **UWF** – for Programs files
    - 2.3.3. **SuperBrain** - for all SuperBrain specials. In this folder the following folders will be created:
      - 2.3.3.1. **bmp** - for “.bmp” files as graphic display on the LCD screen.
      - 2.3.3.2. **obj** – for internal use of the program. Includes “.obj” files.
      - 2.3.3.3. **ALL\_OBJ** - for internal use of the program. Includes “.obj” files.
3. Programming.
  - 3.1. After running *WinBAK\_SBEx1* :



Select the project file you need, or create a new project file.

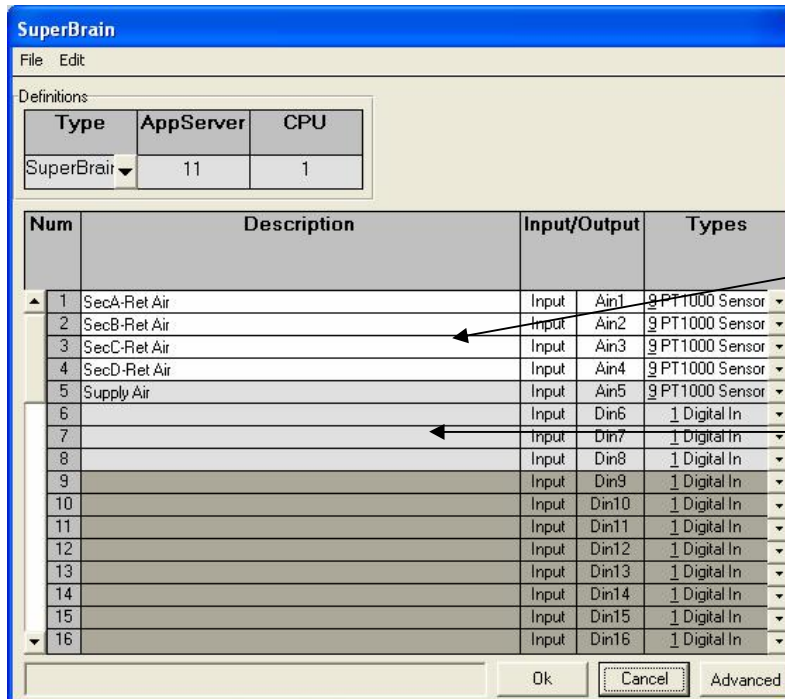


3.2. The following window will appear:



In this window – **program** the WinBak according to requirements.

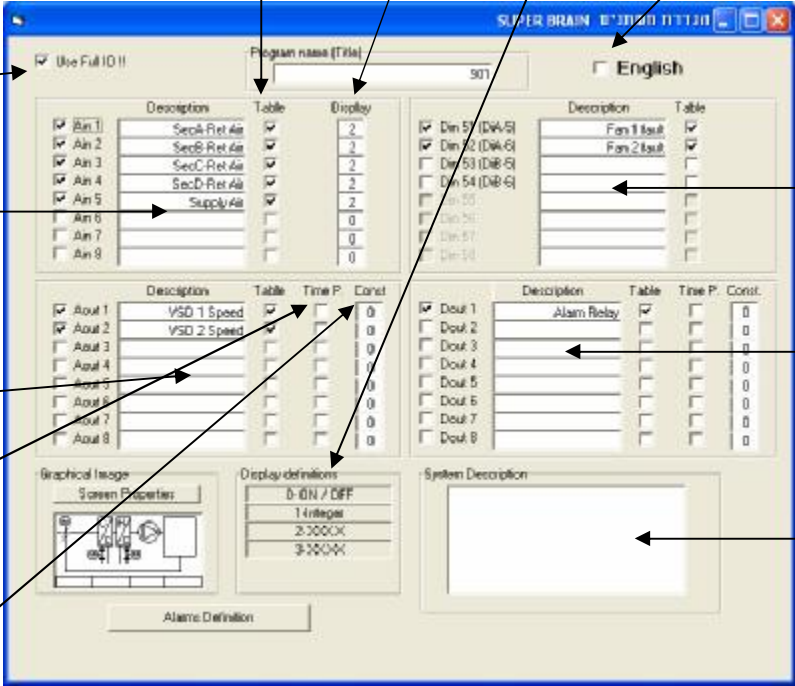
3.3. IO setting and descriptions:



Descriptions and settings for System A (white areas)

Using the gray areas will set the controller to System A only, using all the controller's IO

3.4. SuperBrain specials: select :””.



Multi lingual options for future use. Meanwhile, type all properties in **English mode and non-English mode**.

Valve display, according to "Display definitions"

If checked, IO will be displayed on TEXT – mode display

Sets the program as a "System A" only.

IO descriptions

IO descriptions

IO descriptions

Enable *Time Program* for an Output

Sets a *constant* time program number for Output

IO descriptions

System descriptions will be displayed on the controller by pressing "help" on the controller.

**English**

Description	Table	Display
<input checked="" type="checkbox"/> Ain 1	SecA-Ret Air	2
<input checked="" type="checkbox"/> Ain 2	SecB-Ret Air	3
<input checked="" type="checkbox"/> Ain 3	SecC-Ret Air	2
<input checked="" type="checkbox"/> Ain 4	SecD-Ret Air	2
<input checked="" type="checkbox"/> Ain 5	Supply Air	2
<input type="checkbox"/> Ain 6		0
<input type="checkbox"/> Ain 7		0
<input type="checkbox"/> Ain 8		0

Description	Table	Time P	Const
<input checked="" type="checkbox"/> Aout 1	VSD 1 Speed		0
<input checked="" type="checkbox"/> Aout 2	VSD 2 Speed		0
<input type="checkbox"/> Aout 3			0
<input type="checkbox"/> Aout 4			0
<input type="checkbox"/> Aout 5			0
<input type="checkbox"/> Aout 6			0
<input type="checkbox"/> Aout 7			0
<input type="checkbox"/> Aout 8			0

Description	Table	Time P	Const
<input checked="" type="checkbox"/> Din 51 (DA-5)	Fan 1 fault		
<input checked="" type="checkbox"/> Din 52 (DA-6)	Fan 2 fault		
<input type="checkbox"/> Din 53 (DA-6)			
<input type="checkbox"/> Din 54 (DA-6)			
<input type="checkbox"/> Din 55			
<input type="checkbox"/> Din 56			
<input type="checkbox"/> Din 57			
<input type="checkbox"/> Din 58			

Description	Table	Time P	Const
<input checked="" type="checkbox"/> Dout 1	Alarm Relay		0
<input type="checkbox"/> Dout 2			0
<input type="checkbox"/> Dout 3			0
<input type="checkbox"/> Dout 4			0
<input type="checkbox"/> Dout 5			0
<input type="checkbox"/> Dout 6			0
<input type="checkbox"/> Dout 7			0
<input type="checkbox"/> Dout 8			0

Display definitions

System Description

### 3.5. Alarm definitions:

Alarms Definition

Alarm#

Up to 60 alarms for program. Defines the text displayed on LCD screen in case of alarm.

Multi lingual options for future use. Meanwhile, type all properties in **English mode and non-English** mode.

Alarm#	Alarm Name	Number
31	High Temp Alarm	1
32	SecA-Temp Alarm	2
33	SecB-Temp Alarm	3
34	SecC-Temp Alarm	4
35	SecD-Temp Alarm	5
36	Fan 1 fault	6
37	Fan 2 fault	7
38		8
39		9
40		10
41		11
42		12
43		13
44		14
45		15
46		16
47		17
48		18
49		19
50		20
51		21
52		22
53		23
54		24
55		25
56		26
57		27
58		28
59		29
60		30

English

Copy From WinBak

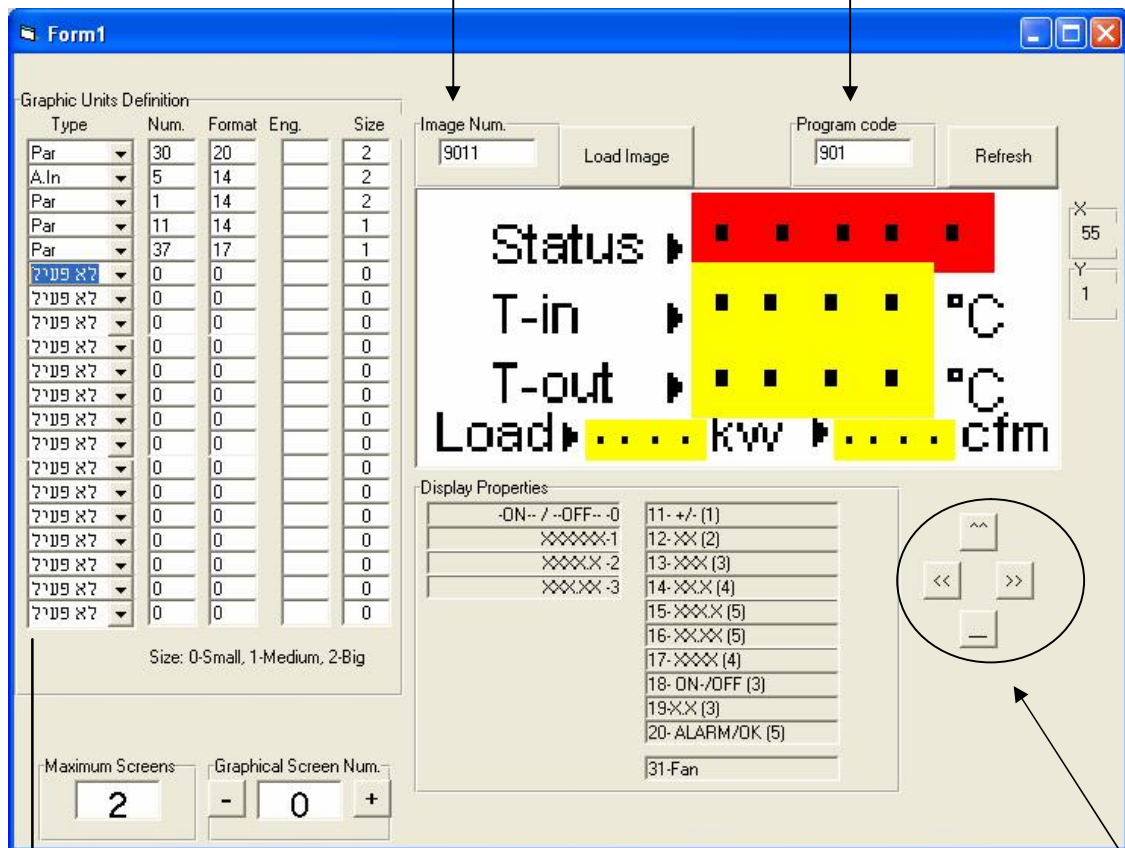
An option to copy alarms text from WinBak. This will delete all existing alarms descriptions.

### 3.6. Graphic settings

Screen Properties :

Sets a bmp file as background for graphic display. It refers to files in bmp folder

Setting the program ID number. **Very important.**



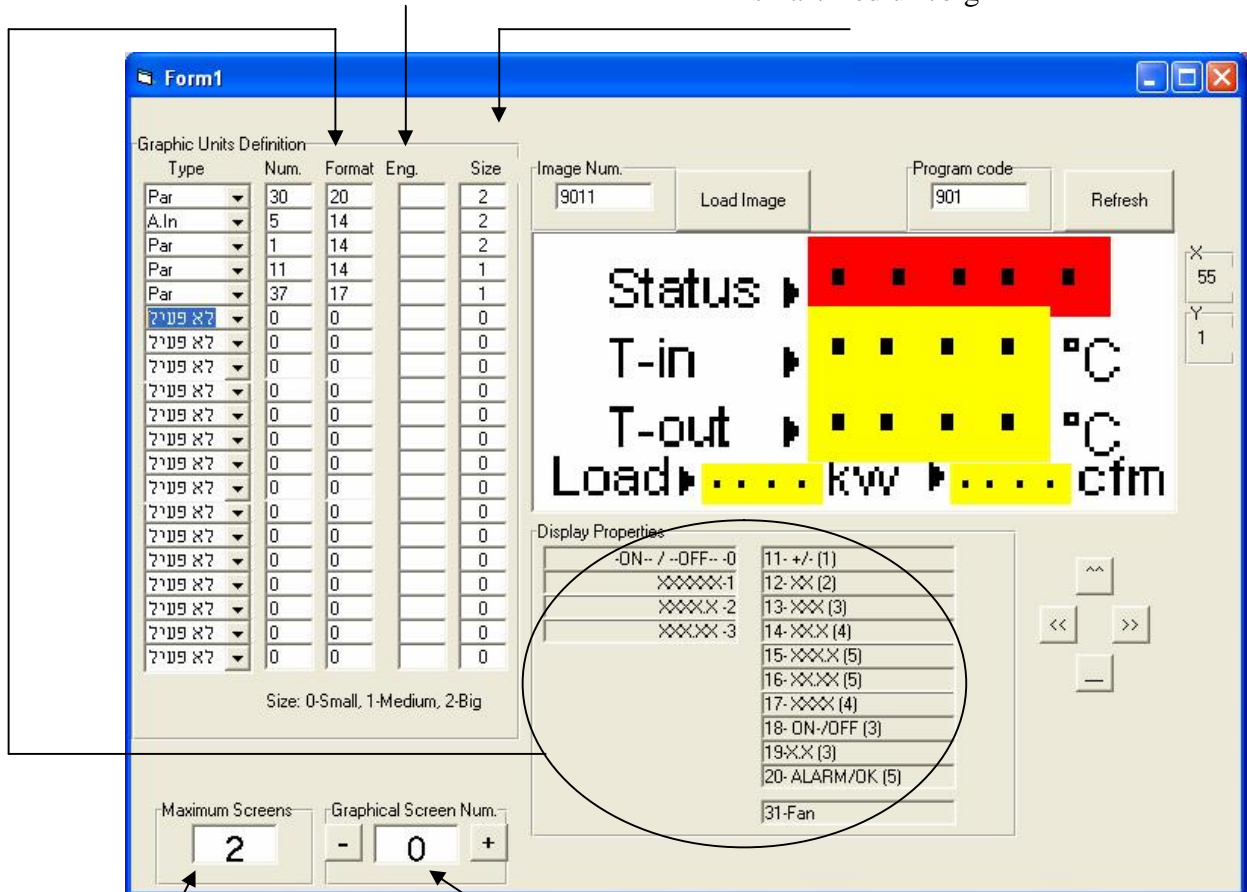
- לא פנוי
- A.In
- A.Out
- D.In
- D.Out
- Par

- Select type of control to display, the number (par# / Ain# / Dout# / etc...) and the format, according to display properties.

Place the point in the required location with the mouse. Use fine locating if necessary.

Engineering units.  
Using this, will **blink**  
the value / Eng.-Units on  
the display.

Size of point:  
0/1/2 =  
small/medium/big




Quantity of graphic screens to  
application (maximum 9 screens).

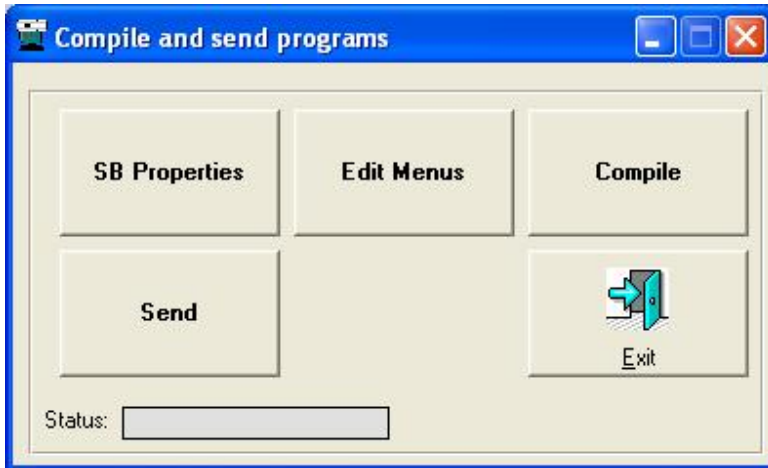
The current graphic screen:  
Graphic screens are numbered from 0 to 8.

3.7. After setting all, **minimize windows and save changes** from WinBak program

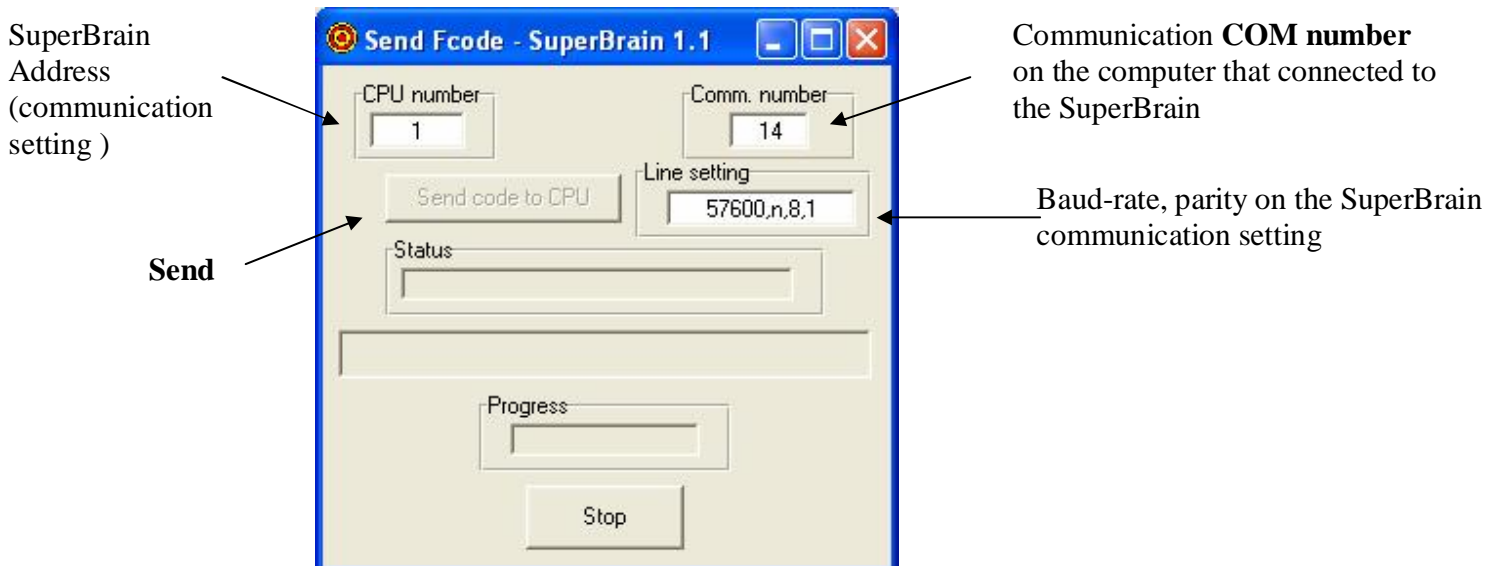


4. Compile and send to controller.
  - 4.1. program send.

Press the compile button in WinBak .

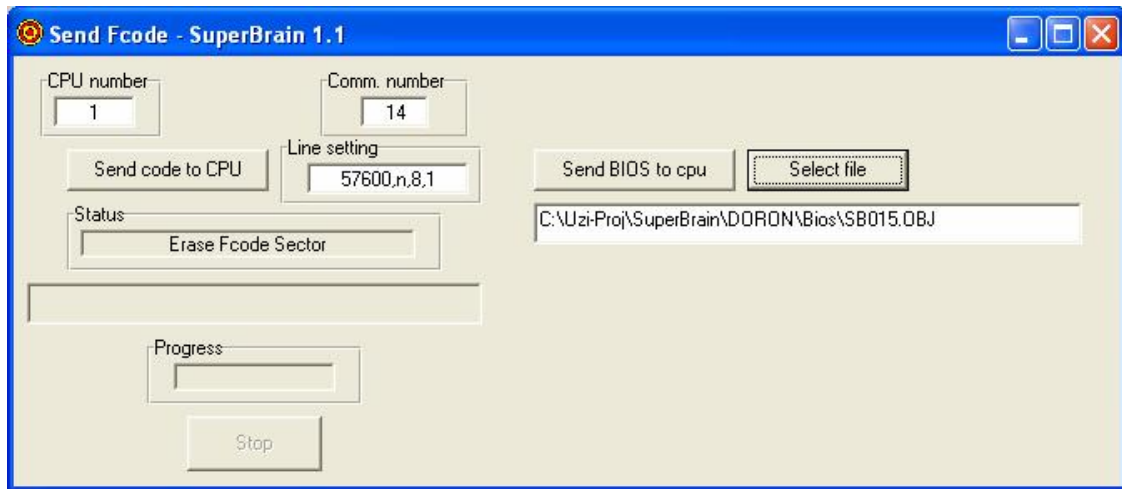


- SB Properties – links to window in section 3.
- Edit menus – not in use.
- Compile – compiles the current project and file and creates an *obj* file in the obj folder. The **status** of compilation will be displayed. If there are no errors the indication will be: **EOP**.
- Send - **Sending all projects** in the working folder to the controller.





#### 4.2. BIOS/Firmware send.



Stretch the Send window, select the BIOS file(“Select File”) and send to controller (“Send BIOS to cpu”).

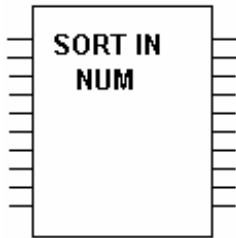
#### 5. Known Beta-version Bugs.

- 5.1. Dout numbering (to insert Dout# choose Dout and select the equivalent Aout# from the IO list)
- 5.2. Mouse hover yellow description on IO incorrect.
- 5.3. Some Hebrew indications.
- 5.4. “System Description” - text align.
- 5.5. No automatic save for special SuperBrain settings.
- 5.6. Communication settings for “send” are not saved.

---

## Sort-IN

Taking the input values and sort them from lowest to highest.  
Num – amount of inputs to sort.



---

## FLS

Flood sensor – reading the value and from the sensor and according to it turns on the correct output of the function.

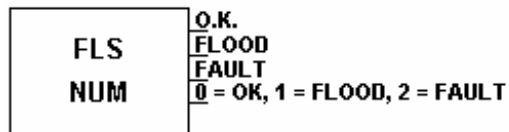
Num – number of input connected.

2.5-5.5 volt - flood (OUT NO. 2)

6-10 volt - ok (OUT NO. 1)

All the other voltage - fault (OUT NO. 3)

The value (OUT NO. 4)



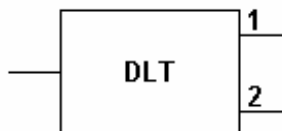
---

## DLT

Delivers to the outputs the delta between readings of the input per one round of the program.

Output no. 1 gets the positive and the negative delta value.

Output no. 2 gets only the positive delta value.





## **FOUT**

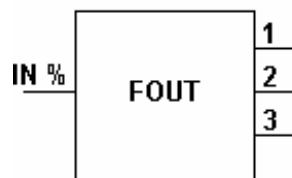
Flout engine.

Inputs:

1. The demand value 0-100 %
2. Period time in sec.
3. Reset
4. Time in minuets for auto reset.

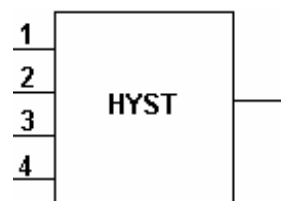
Outputs:

1. Closing command.
2. Opening command.
3. Location (calculated).



## **HYST**

The same as HYS, except for input no. 4 – time in seconds in which the change in the inputs should remain before changing the output.



## SMS

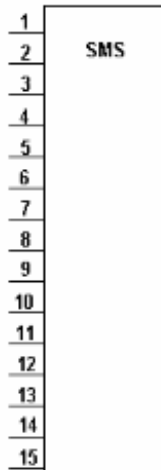
Used for sending SMS messages from controller (using GSM modem).

Inputs 1- 12 - the numbers of alarm to be sent.

Input 13 - zip code.

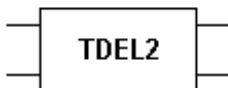
Input 14 - phone number to send to.

Input 15 - enable function.



## TDEL 2

The same as time delay except for additional output no.2 which shows the counting of time already past.



## Toff

When input number 1 turn from 1 to 0, the output will change after delay value in seconds connected to input number 2.

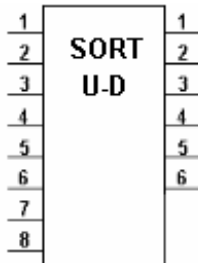


### **Sort Up – Down, Down – Up**

Inputs 1-6 : the values.

Input 7 : if equal 0 sorts from high to low, if equal 1 sorts from low to high.

Input 8 : The function sorts per each rising of input 8 from 0 to 1, if the function remains 0 or 1 no sorting is done.



---

### **XBTN**

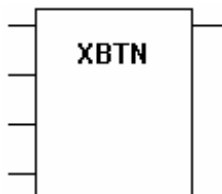
External button:

Input 1 – input from time schedule SST, when the input is 1 the output gets 1 as well.

Input 2 – pushbutton input, each pulse changes the output (during time schedule as well), depends on input 4.

Input 3 – number of parameter that can be used as a pushbutton for the function or to be affected by it's output.

Input 4 – time in minutes to stay activated after a pushbutton was pushed.

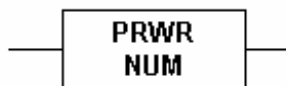


---

### **PRWR**

Combination of PROR and PROW.

NUM – Number of the Modbus row which register wanted to be read or write to.



## AVRX – average

Inputs :

1. The value.
2. N – seconds.
3. Reset

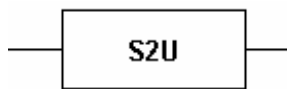
Outputs:

1. Every N seconds gets the average of the value.
2. Every second gets the average for last N seconds.
3. Shows how N time elapsed.



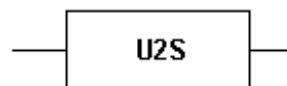
## S2U

Turns Signed to Unsigned.



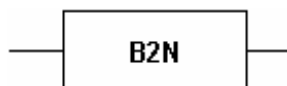
## U2S

Turns Unsigned to Signed.



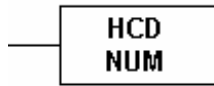
## B2N

Turns BCD format value to decimal.



### **HCD - Hold Count DIN**

Used with function CNTD , when the input equal 1, the count stops.  
NUM – Number of input been used.



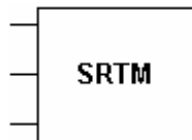
### **SRTM – step relay time**

Used with function SROt in order to define the next parameters (if not used as default in controller)

Input 1 - The size of the pulse in seconds.

Input 2 - Time for output status recheck.

Input 3 - Number of trials to reach the desirable status.



### **SREL**

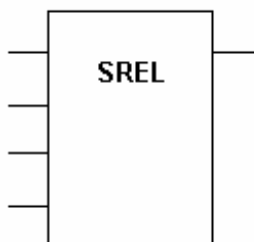
Time schedule with external button out of schedule.

Inputs:

1. First time schedule.
2. Second time schedule.
3. External button.
4. Time in minutes for activation out of schedule (using external button).

Output:

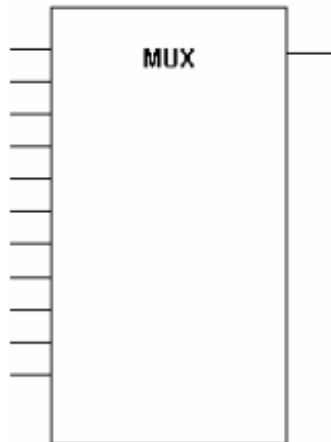
1. Gets value of 1 if input 1 or 2 is ON or if input 3 is ON out of schedule.



---

## **MUX**

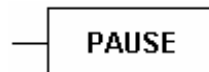
The output will get the value of an input which number is connected to input no. 11 .



---

## **PAUSE**

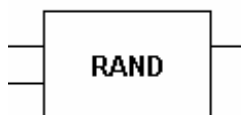
Pause the program for amount of seconds connected to the input.



---

## **RAND**

At each rising of input no. 1 from 0 to 1 (for each pulse up - if the function remains 0 or 1 function disabled). The output will receive a random number from 0 – N which connected to input no. 2 .



---

## **TAOZ**

For Israeli use only.

---

## **FORCED**

Checks whether there is forced IO in the controller.

Output NO. 1- Get value of 1 whether An In forced.

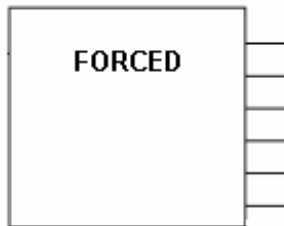
Output NO. 2 - Get value of 1 whether An Out forced.

Output NO. 3- Get value from 1-16 for the number of An In that forced.

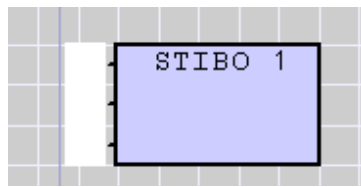
Output NO. 4 - Get value from 17-32 for the number of An In that forced.

Output NO. 5- Get value from 1-16 for the number of An Out that forced.

Output NO. 6 - Get value from 17-32 for the number of An Out that forced.



## **STIBO**



Allows sending data to a slave controller at “no come” communication status.

The internal number is a slave cpu address.

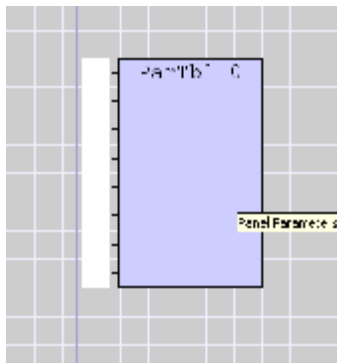
Input No.1 – The value to be send

Input No.2 – Target file number

Input No.3 – Target parameter number

## ParTbl

Allows choosing which parameters are accessible by Unisense panel clock button.



The internal number is the parameter file number.

Inputs 1 – 2: Allows to define a group of parameters input 1 is a parameter number to start from up to parameter number as appears in input 2.

Inputs 3, 4, 5, 6 – allow access to single parameters.

Inputs 7,8 – allows to define the presentation format:

If input 7 = 0 or not connected, then parameters in inputs 1-2 will remain in regular format, If input 7 = 2, then parameters in inputs 1-2 will appear in HH:MM format.

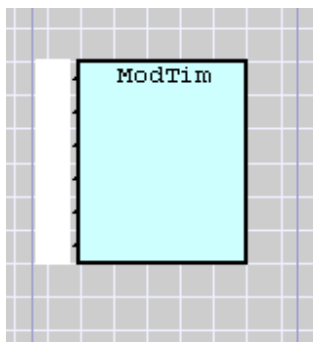
If input 8 = 0 or not connected, then parameters in inputs 3,4,5,6 will remain in regular format, If input 8 = 2, then parameters in inputs 3,4,5,6 will appear in HH:MM format.

The table can present up to total amount of 64 parameters.

---

## ModTime

Allows defining the intervals and number of retries in modbus commands.



Input 1: Number of retries when **reading** data.

Input 2: The wait before time out (give up waiting) when **reading** data.

Input 3: Time interval when **reading** data.

Input 4: Number of retries when **writing** data.

Input 5: The wait before time out (give up waiting) when **writing** data.

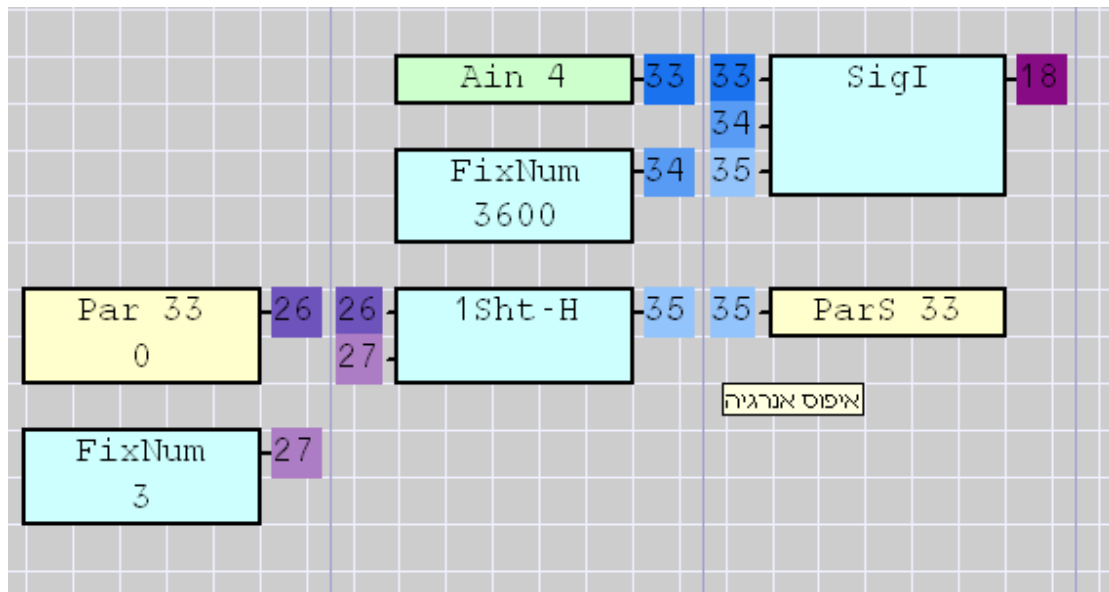
Input 6: Time interval when **writing** data.

All the time intervals are in msec.



## SigI

Running average sampling and storing.



Input 1: The input for sampling.

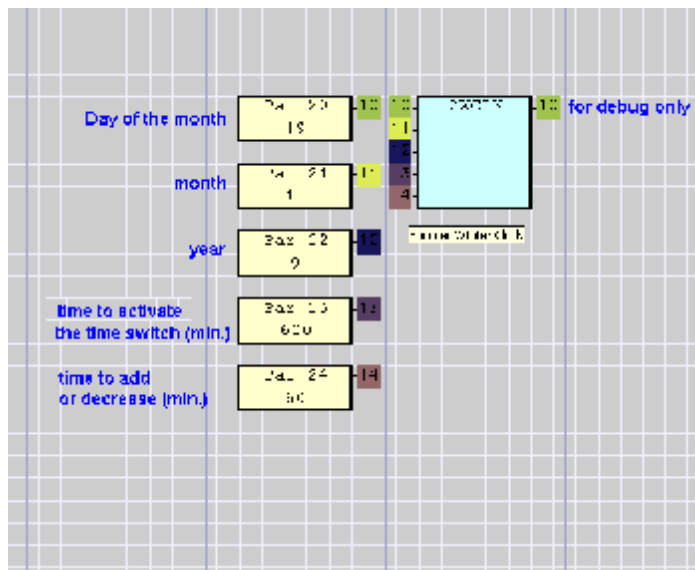
Input 2: Sampling frequency in seconds.

Input 3: Reset the stored data.

Output : Stored data.

## SWCLK

Summer Winter Clock change – changes the CPU's time at desirable date and time.



Input 1: The date to apply the time shifting.

Input 2: The month to apply the time shifting.

Input 3: The year to apply the time shifting (if nothing is connected will happen every year).

Input 4: The time (in min.) to apply the time shifting.

Input 5: The time shifting (in min.).