th-Tune

new firmware 1.1

Programmers' guide

Ver. 4.8

14/02/2013

CAREL

Integrated Control Solutions & Energy Savings

Introduction

This presentation provides some basic explanation to develop an application with one th-Tune display connected to a programmable controller.

The features here described refer to the new firmware version (1.1); the firmware version is displayed at the power on of the display.

As a consequence there are some differences between this programmers guide and the former one (referred to the 1.0 firmware version. There will be also a new version of 1tool but it's not strictly necessary to have it (the new version will improve usability). The references to th-Tune characteristics that changed are displayed with this icon 

The th-Tune can be used with two different types of protocol in the programmable controller; these two connections have different possibilities (indicated with different colors) so the described features will have a correspondent color indication to show if it's available for both options or not 

Topics 1 / 3

1. Main features
   Product overview, applications
2. Applications
3. Display structure and Terminal usage overview
   display area, how to modify status, key function (pre-defined), …
4. How to create new project for th-Tune
   Solution Exp.editing limits, Windows, Properties, Languages, N.of mask
5. Terminal Editor components and Editing
   Display area, Key area, Tab area,
   Hot area: Mode, Fan, Icons, Scheduler, Big/SmallArea.
   How to assign a variable to a ‘function’. Automatic creation of variables, Copy/Move of variable
6. Configuration of communication parameters

Topics 2 / 3

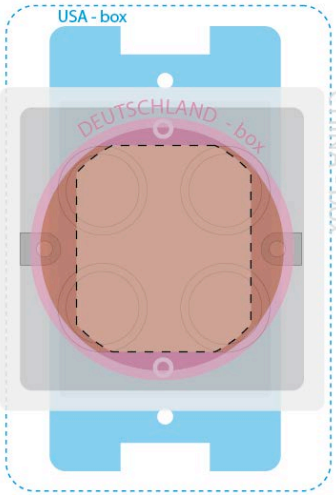
Topics 3 / 3

Main features 1 / 3

- Standard power supply (115..230 Vac) or 24 Vac/Vdc
- Temperature and Humidity measurements
- Internal clock
- Compatible with IT, DE, CN and US flush mounting sockets
- Time schedule: 5+2 or 7 days programs and single day programs (up to 6 time bands per day)
- Communication with the controller:
  - RS485 with "***Modbus master protocol***" blue colour is indicating the protocol in this presentation
    - More terminal/device on the same net
    - Only basic function are available
    - Traditional programming of a modbus device
  - RS485 with "***th-Tune protocol***" orange colour is indicating the protocol in this presentation
    - Only one th-Tune terminal can be connected
    - It'd not possible to connect other devices
    - Available only **FieldBus** port
    - It's "***Modbus master protocol***" managed by BIOS
    - Specific terminal editor in 1tool
- Operating conditions: -10T50° C

Main features 2 / 3

- Easy To use
- User Terminal as well as Commissioning terminal
- Compatible with the most common wall sockets available in the market

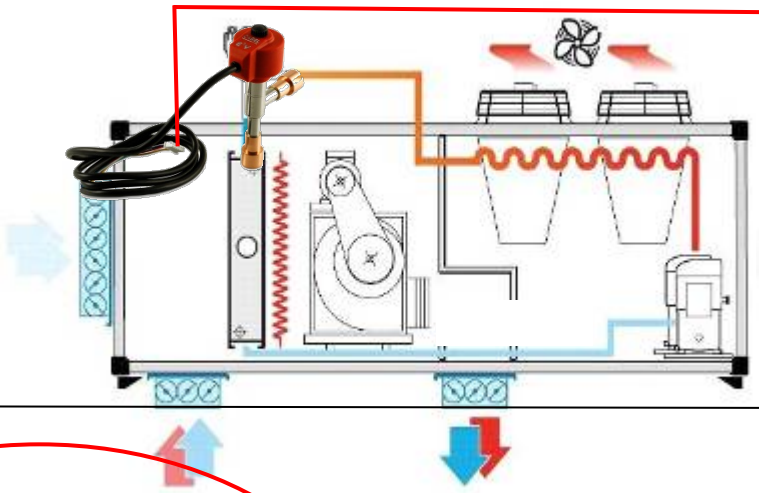


Main features 3 / 3

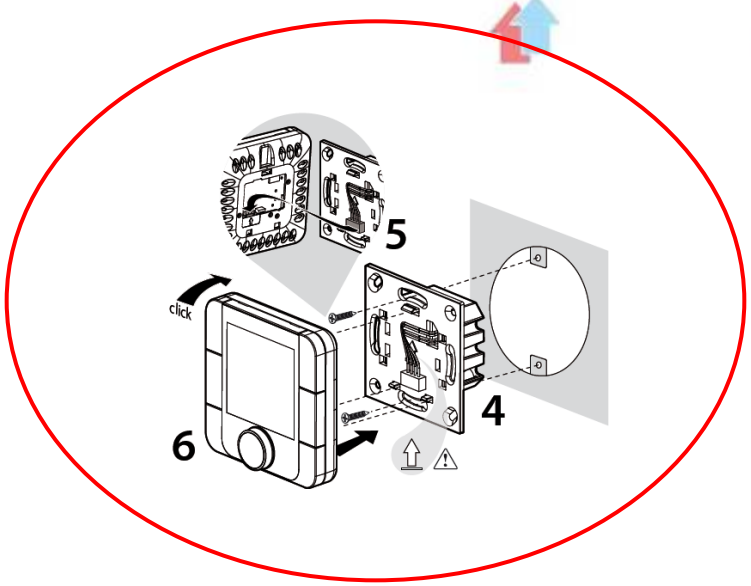
- "Dummy" user interface (predefined keyfunction & navigation must be supported by the controller)
- Can be used together with PGD* terminals
- "thTune protocol" FieldBus serial port or "serial 0"
- Programmable by *1tool*
- Support module available soon
- Supported by: pCO1* (=>2MB flash), pCO3, Supernode, pCO5)
- 1tool version: => 2.0.31 or higher
- BIOS: => 5.11, 04/10/2010
- Firmware: => 1.1

th-Tune **Application:** unit display
Advanced functions

Bringing it on the unit it's possible to see all the parameters with a single display: up to 15 categories of parameters



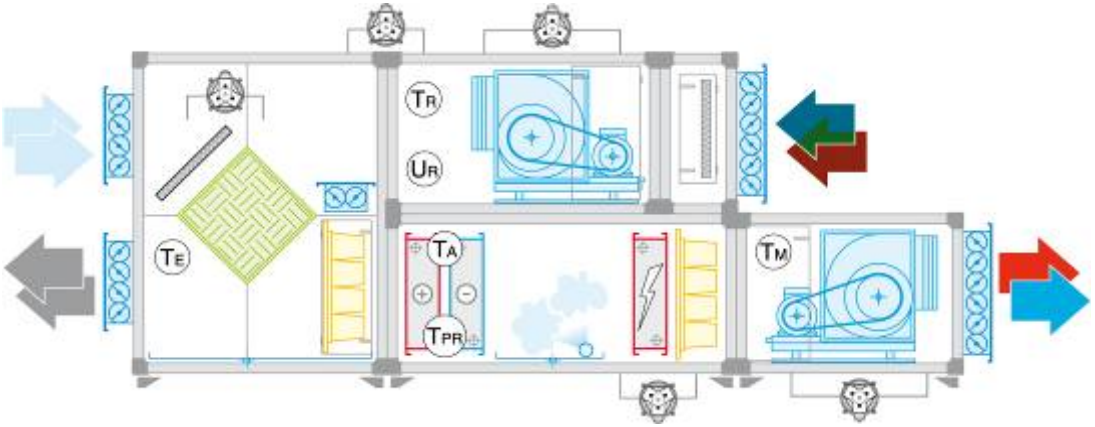
"th-Tune protocol"
on FiledBus port



Room

th-Tune **Application:** user terminal for basic settings

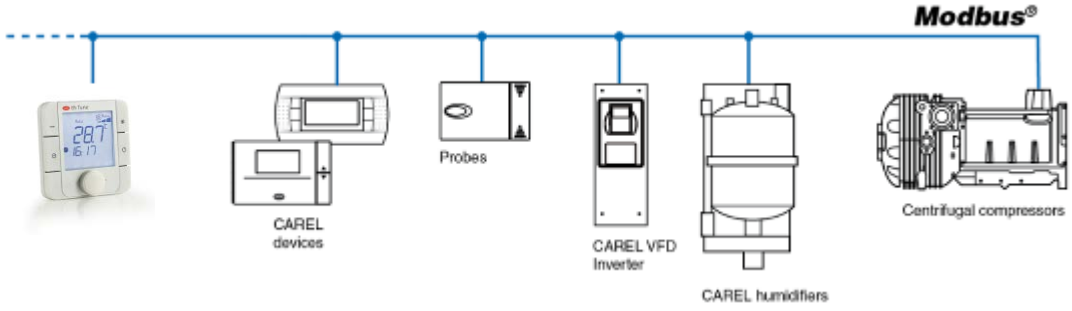
ON-OFF, setpoints, temperature and humidity measurement, function enabling, TIME BANDS messages and icons



Modbus® Master

Using standard modbus only basic functions are available but it's possible to insert more displays and different devices (the limit is speed of information)

Room



Display structure and Terminal usage overview 1 / 3

“Mode” area

“BigArea”
Shows the value of one (fixed) or more variables (the Carousel) and shows the value is being set by the user.

“Time bands”
Enable/disable

“SmallArea”
Shows the value of one (fixed) variable or the time or the acronym of the variable is being set.

“Scheduler” area

“Fan” area

Locked function icon

“Time band” area

“Icons” area

Display structure and Terminal usage overview 2 / 3

“Mode”

When pressing the ‘**Mode**’ button, the symbol relating to one of the 5 modes available will be shown:

- Heating
- Cooling
- Automatic
- Dehumidification
- Water

(these are the most common associated functions to those symbols but the real operation depends on application)

“Clock”

When pressing the button marked by the ‘**Clock**’ symbol, the terminal will switch to time band setting mode. Keeping the button pressed it will be possible to enter in Clock and Time bands settings

“Fan”

When pressing the button marked by the ‘**Fan**’ symbol, the speed bar will show the next level.

“Power On / Off”

When pressing the button marked by the ‘**Power On / Off**’ symbol, the terminal will show ‘OFF’ in the centre of the display.

In special menus (time bands and clock settings, alarm menu, parameters menu is equivalent to ESC function)

“Encoder”

Push and Rotate to change value/parameter

Display structure and Terminal usage overview 3 / 3

“Mode” + “Clock”

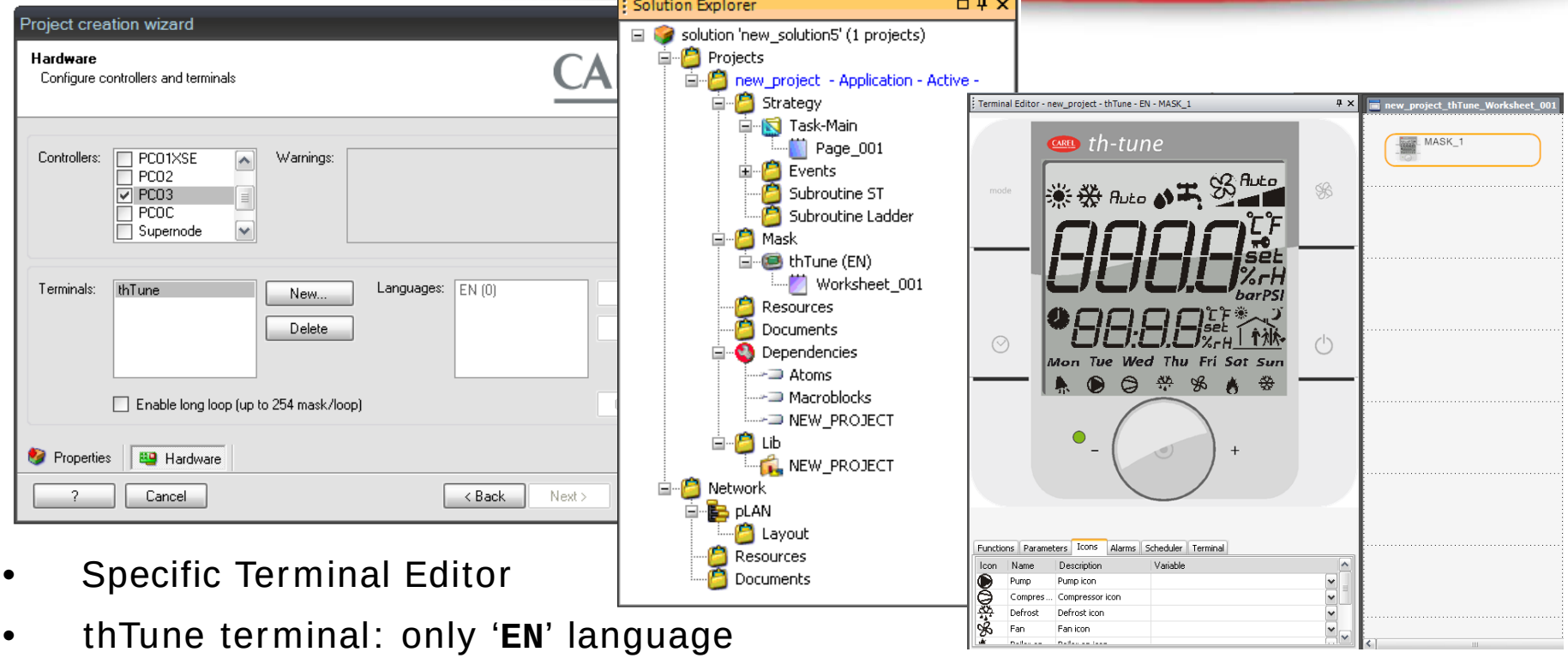
When pressing these buttons together for 5 seconds, if there is at least one active alarm (blinking bell icon) it is possible to enter in Alarm visualization and reset menu. Alarms must be configured in Alarms TAB

Blinking “BELL” icon

“Fan” + “Power On / Off”

When pressing these buttons together for 5 seconds, and then setting the right password, it is possible to enter in Parameter Menu.
Custom Parameters can be configured in Parameters TAB.
There are some fixed th-Tune internal parameters always present

4 How to create new prj for th-Tune (if using th-Tune protocol)



- Specific Terminal Editor
- thTune terminal: only '**EN**' language
- Only 1 mask/terminal
- Most part of editing menu/tools/properties are disabled.

  Example: *Toolbox/NormalMask* only, *Configure Languages*, *Delete*, *unused SpecialFields*, *KeyFunctionEditor*, *ResourceUsedOnBool*, etc.

NOTE: When compiling a *.iup is generated to be downloaded in the controller ONLY if there is no other *.iup

4 How to create new prj for th-Tune (if using Modbus master protocol)

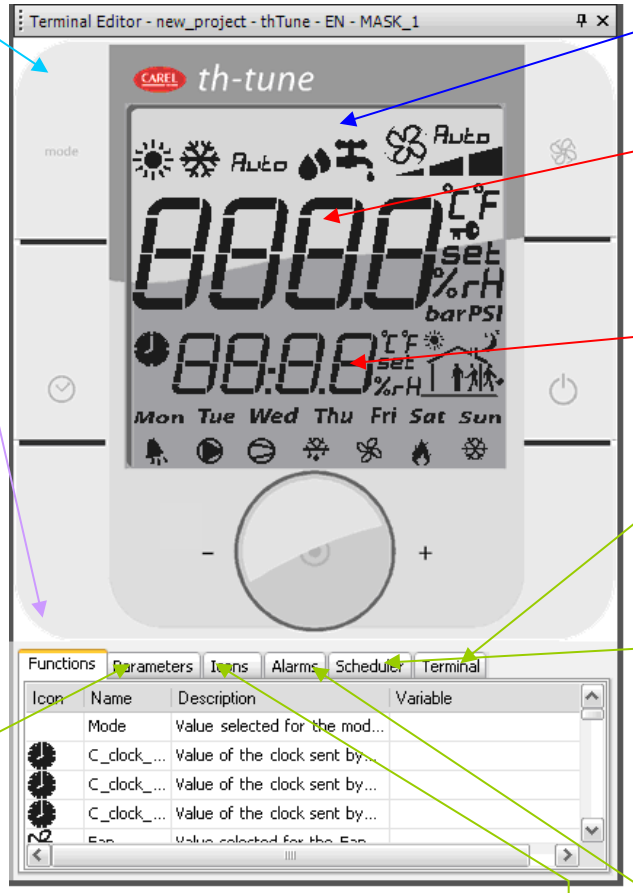
- Check list of variables available in Modbus
- Use de Modbus master DEMO to understand how it works
- Develop your own application integrating the th-Tune as a simple Modbus device
- Please note that some integer bit field variables in the th-Tune editor must be sent as separate coils in modbus protocol (in th-Tune protocol the bios is automatically doing that)
- Make sure that you set the right variables for configuration (see section 7)

Terminal Editor components and Editing 1 / 2

'Button area'
of the terminal.

'Functions tab'
This shows the graphic symbols for the 5 functions ('Mode', 'Fan', 'Clock', 'Power' & 'Alarms') on the display that can be associated with a variable dragging it from the 'Variable List'.

'Parameters tab'
This lists the parameters designated to be manually set by the user, dragging them directly from the 'Variable List'.



'Icons tab'
This lists all the graphic symbols relating to the functions on the display that can be associated with a **variable** by dragging it from the '*Variable List'*.

Mouse 'sensitive area'

'LED display 1' (BigArea)
shows the value is being set by the user.

'LED display 2' (SmallArea)
shows the acronym of the variable is being set.

'Terminal tab' Address of the terminal (**only one**, must match with the one set on th-Tune)

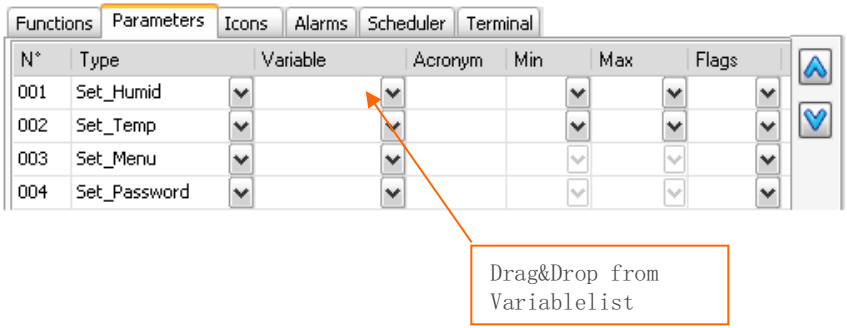
'Scheduler tab'

Not USED it will be deleted in next versions of 1tool
The scheduler now is managed internally by th-Tune

'Alarms tab'
This lists the variables relating to the alarms used in the application.

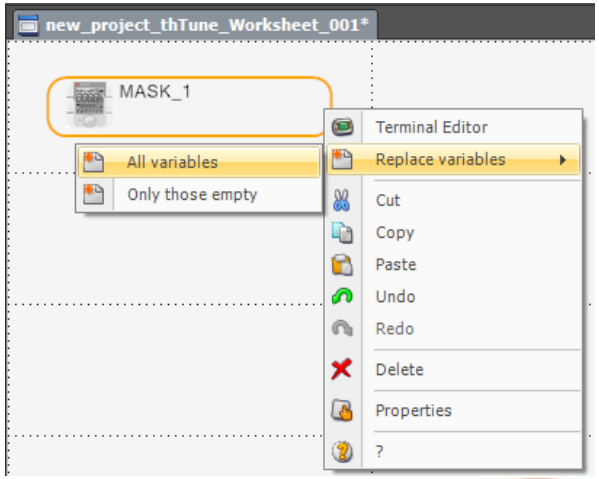
Terminal Editor components and Editing 2 / 2

- How to associate a *Variable* to a *Symbol/Function/Status/Variable*
  - drag&drop from ***Variablelist*** to *Symbol/Function/Status/Variable or to Listview*
  - Type the variable in the **Listview**



- Shortcut
  - MOVE and CTRL+MOVE(=copy) available
  - CTRL+X/C/V not available

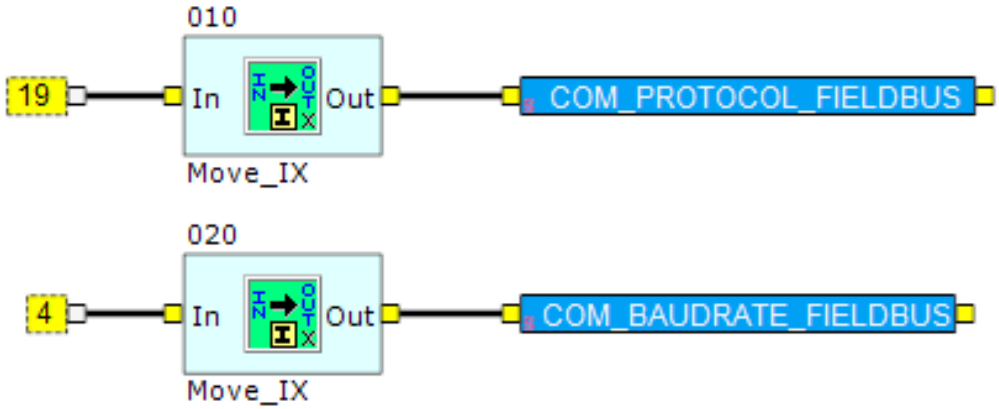
- Automatic creation of support variables: Right-click the mask, *Replace variables…*



- *th-Tune* support module and demo (preliminary version is available) is recommended as a possible starting point for new projects

Configuration of communication system variables

- Inside project you must set this comunication parameters on FieldBus port
  - COM_PROTOCOL_FIELDBUS=19 (th-Tune protocol)
  - COM_BAUDRATE_FIELDBUS=4 (19200baud)



NOTE

- '***th-Tune protocol***' is avaible on FieldBus port but also on serial 0 (pLAN for small lenghts of the cables). For serial line 0 you'll have to set the specific system variable
- '***th-Tune protocol***' support only **one** th-Tune terminal
- Comunication between pCO and terminal is managed automatically by BIOS
- 1tool provides to user a quickly and simple graphical interface to manage th-Tune variables read/written by BIOS

6

Configuration of communication th-Tune parameter

Make sure that baudrate and address on the th-Tune are correct: see section 18 for th-Tune parameters.

Terminal configuration
configuration and status variables

There are some variables indicating the status and the configuration of the terminal that must be used for basic settings of th-Tune behaviour. Make sure that you set these variables in the right way to configure the main features of the terminal and then design the functions with the following section 7

Terminal configuration
Cfg_flags

- Use **Cfg_flags** variable to configure the terminal

BIT	Function	Description
0	Bypass “INIT funtion”	This flag tells to terminal to bypass the initialization phase used for parameters and alarms in “th - Tune protocol” MUST be used when communicating in **MODBUS** otherwise it’s not necessary
3	measure unit of temperature displayed	This flag tells to terminal which unit of measure is used to represent the temperature (° C or ° F), so as to show the corresponding symbol (displays symbol in Big/SmallArea with conversion). 0 = ° C 1 = ° F
4	measure unit of pressure displayed	This flag tells to terminal which unit of measure is used to represent the pressure (BAR or PSI), so as to show the corresponding symbol (displays only symbol in SmallArea, without conversion). 0 = BAR 1 = PSI
5	Load time events	This flag tells to terminal to copy in the scheduler the BMS variables for time bands, that are available in R/W mode only in Modbus protocol
9	Hours display	0 to display hours of the internal clock, 1 to display hours sent by pCO (see “**Internal clock**”)
10	Time band 1	If 1, the time band 1 isn’t used and displayed
11	Time band 2	If 1, the time band 1 isn’t used and displayed
12	Time band 3	If 1, the time band 1 isn’t used and displayed
13	Time band 4	If 1, the time band 1 isn’t used and displayed
14	Time band 5	If 1, the time band 1 isn’t used and displayed
15	Time band 6	If 1, the time band 1 isn’t used and displayed
other	Reserved	Not used

NOTE:
when using Modbus protocol, it’s necessary to write directly the coils for each specific enabling

Terminal configuration
Status_Flag

- **Status_flags** variable
  - status register of the terminal used to provide general information.

BIT	Function	Description
0	Heartbeat	Managed by pCO BIOS. Terminal sets to 1 this bit after a power off, or when the th-Tune doesn't detect any valid interrogation from a Modbus supervisor to a Modbus slave in the same network within 30 seconds (terminal displays "Cn"). When BIOS sees to 1 this bit, it sends all variables to terminal and sets this bit to 0 (so all variables are aligned).
1	"Time bands" enabling	The state of this flag is changed by terminal with a short-press of "CLOCK" key, user in this way can enable/disable "Time Bands". The display shows this icon when "Time Bands" are enable. (see section **12**)
4	Custom function	If =1 the "custom function" is active (see section **11**)
5	Internal clock (RTC)	=1 (internal clock always present)
7	External-Internal Scheduler	This flag is set to 1/0 by application to configure 0= "Time bands" are managed by the pCO that must have the scheduler in its memory, change the setpoint, the current time band and the "Power" variable 1= "Time bands" are managed by the th-Tune that changes the setpoint. Automatically with this option the set editing function is disabled writing the specific bit of "key_enable_mask" variable, but it's always possible to enable it from application
15	Terminal online/offline	Application can only read bit15: 0=terminal online, 1=terminal offline. This bit is set to 1 by bios when using th-Tune protocol after an interrogation that receives no answer from th-Tune

Terminal configuration

- Example of using cfg_flags

How to use **Cfg_flags** variable

- `Bit3= 1` to display ° F symbol in Big/SmallArea, and the "**Temperature**" is already converted by terminal
- `Bit4= 1` to display PSI symbol only in BigArea, is necessary to convert only "**Pressure**" before displaying it on the terminal

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												1	**1**			

NOTE: when using Modbus protocol it's necessary to write a specific coil

New management at the power on of th-Tune

- At the power ON all the keys are disabled and the Small area is showing the label “init”: it will be possible only to enter in parameters menu to set the ADDRESS

- this phase is used in “th-Tune protocol” to send the acronyms of parameters; when the sending acronym phase is ended it goes out from INIT phase

- to use the th-Tune in Modbus master it is necessary to set to “1” the bit 0 of cfg_flags (see correspondent modbus coil) to make it go out of “init” phase

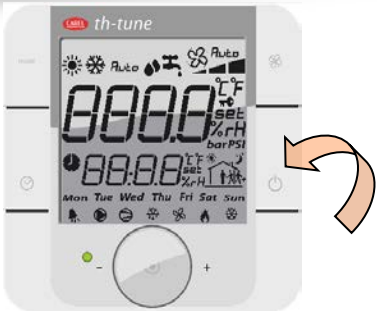
- during this phase anyway the th-Tune is communicating so it can be used for initialization

- It is possible to set a “th-Tune internal parameter” (password 22) to automatically bypass this phase. The parameter is P_In and the default value of “0” must me set to “1”

- the offline of th-Tune (showing “Cn”) is now after 30 seconds and it’s counted from the last valid interrogation on the network; asd a consequence if the master is sendin valid modbus messages to another controller the th-Tune is ONLINE

State of display

How to manage th-Tune display state:



- Use "**Power ON/OFF**" button manage state of display
  - Press for 3s "Power ON/OFF" to switch on/off th-Tune display
  - **Power** variable is related to "**Power ON/OFF**" button state

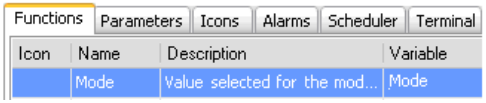
- Use **Power** variable of "Functions" tab or correspondent modbus variable
  - To read display state (i.e. set by user or by scheduler***)
  - To force display state (i.e. set by application)
  - Assumes only 4 value, see table below

Value (Dec.)	Description
0	**Display OFF** The display is in OFF state and shows: - in BigArea "OFF" - in SmallArea the variable set (see SmallArea variables)
1	**Display ON** The display is active and show varaibles set in Big/SmallArea
2	**Display OFF from time band** In BigArea "OFF" and in SmallArea shows the variable set (see SmallArea variables) and
3	**Display OFF** Display is without no symbols and in OFF state

*** see section 13

How to change “Mode” 1 / 2

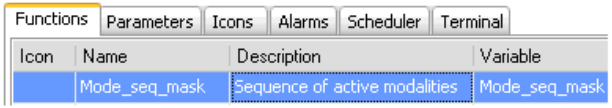
- The “Mode” area includes **several symbols**: heating, cooling, etc
  - To change mode press the ***Mode* button** or use ***Mode*** variable



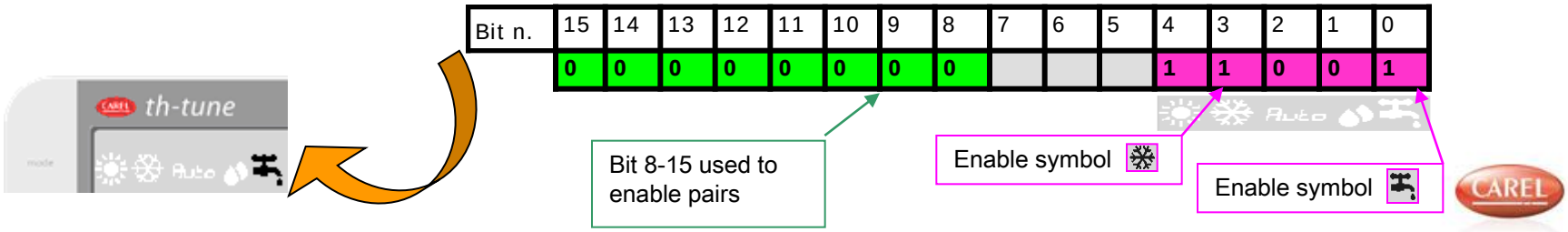
- Current mode (symbol/s) is indicated by variable ***Mode*** (**Read / Write**)
- The symbols can be turn on **separately** or in **pairs** (, , etc)

Example

- The symbols (and pairs) must be **configured** before being used with **Mode_seq_mask** variable



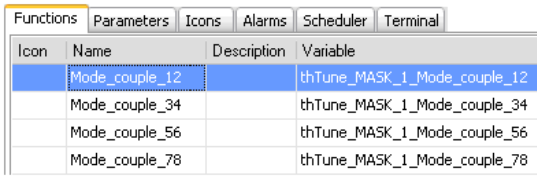
- **Mode_seq_mask** enable the symbols can be turn on **separately** or in **pairs**
  - `Bit0-4: enable individual symbols`
  - `Bit8-15: enable pairs`
  - **`Mode_seq_mask`** `= 0000 0000 0001 1001 bin = 25 dec`



How to change “Mode” 2 / 2

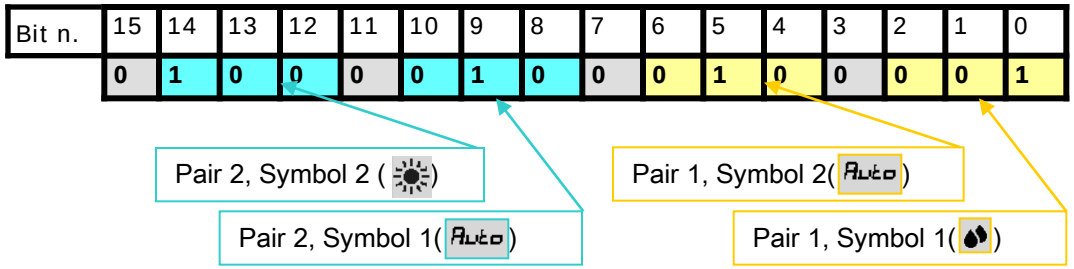
Pairs of symbols

- The **pairs of symbols** must be **configured** before being used (max 8 pairs)
- The pairs of symbols are configured with variables ***Mode_couple_12 / 34 / 56 / 78***
- Value to use inside ***Mode_couple_12 / 34 / 56 / 78***
  - H_20 = 0 dec = 000 bin
  - Dedum = 1 dec = 001 bin
  - Auto = 2 dec = 010 bin
  - Cool = 3 dec = 011 bin
  - Heat = 4 dec = 100 bin

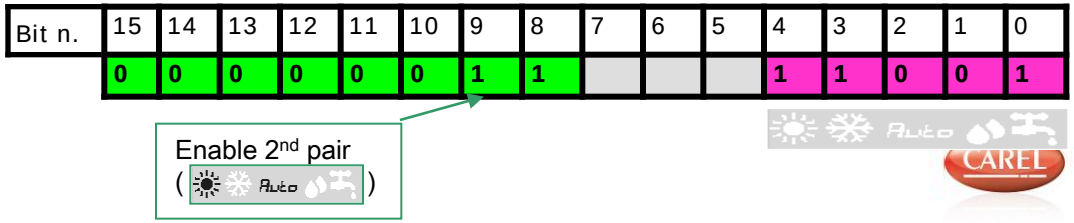


Example

- How to configure two pairs of symbols
  - **Mode_couple_12** = 0100 0010 0010 0001 bin = 16929 dec
  - Pair2=Heat+Auto
  - Pair1=Auto+Dehum



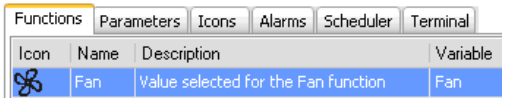
- Symbols/Pairs can be **enabled** with variable **Mode_seq_mask**
  - **Mode_seq_mask** = 0100 0011 0001 1001 bin = 17117 dec
  - Bit0-4: enable individual symbols
  - Bit8-15: enable pairs



How to change “Fan”

- The “Fan” area includes **several symbols**: Fan_Speed1, Speed2, Speed3, Auto
- To change the *Fan status* press the ***Fan* button** or use ***S_Fan*** variable

```
  0: all OFF
  1: Fan_Speed1
  2: Fan_Speed1 + Speed2
  3: Fan_Speed1 + Speed2 + Speed3
255: Fan_Speed1 + Speed2 + Speed3 + Auto
```



- Current ***Fan status*** is indicated by variable ***S_Fan*** (**Read / Write**)
- It is possible to **override** the fan status by using the var **Fan_control_override**

with this variable it’s possible

- Overwrite one of the speed (whatever the value of fan variable is)
- Overwrite AUTO icon (off, blink)
- Overwrite FAN+speed 1 icon
- Make the fan speed request by key vary between blinking status, while is the application to confirm the value and stop the blinking

Note: “Fan” and “Speed1” ( ) are joined together: you can’t use this features to override only “Fan” symbol

Fan_control_override: details

Bit	Description
0,1	Specify the type of SPEED symbols override inside “Fan area”: 00= *OFF*, 01=*Speed1*  , 10=*Speed2*  , 11=*Speed3*  *Note:* S_Fan variable is not affect by this selection: pressing Fan key it possible modify “S_Fan” value in “Fan area”
2	Enable override of the SPEED symbol: BIT 1-2 are used
3, 4	*“Speed1” icon* override: 00= no effect; 01= always OFF; 10= always ON;11= always Blink 
5, 6	“*AUTO*” icon override: 00= no effect; 01= OFF; 10= ON;11= Blink 
7	“Fan” area enable if = 1 “Fan” area disable (all symbols forced off) 
8	Fan Blink request (when this bit is set to “1”, the pressing of fan key makes blinking the selected speed and make it to vary between admitted values. *Note*: after pressing fan key the BIT 9 is set to 0, then application must set BIT 9 =1 to stop blinking)
9	Fan blinking reset: works only if BIT 8 is set to “1”
10	Disable *OFF* selection (no fan speed); when pressing the Fan key, the “S_Fan” variable cannot assume 0 value and it’s not possible to visualize no fan icon
11	Disable *Speed1* selection; when pressing the Fan key, the “S_Fan” variable cannot assume 1 value and it’s not possible to visualize just *Speed1*
12	Disable *Speed2* selection; when pressing the Fan key, the “S_Fan” variable cannot assume 2 value and it’s not possible to visualize just *Speed2*
13	Disable *Speed3* selection; when pressing the Fan key, the “S_Fan” variable cannot assume 3 value and it’s not possible to visualize just *Speed3*
14	Disable *AUTO* selection; when pressing the Fan key, the “S_Fan” variable cannot assume 255 value and it’s not possible to visualize *Speed3 + AUTO*
15	Not used

How to override “Fan” area

Example 1: overwrite

1. Enable override “Auto” symbol in blinking mode
2. Enable override “Speed 3” symbol
3. “Auto” symbol Blinking : bit 5-6 → 11
4. “Speed” symbols = [redacted] : bit 0-1 → 11
5. Speed overwrite enable: bit 2=1

fan_control override

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0	0	0	0	0	0	0	0	1	1	0	0	1	1	1

The fan variable keeps its value [redacted] and “Auto” is blinking.

Example 2:

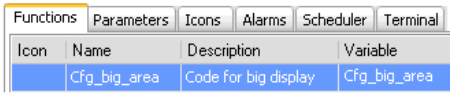
1. Enable blinking request: bit 8=1
2. User changes fan speed with FAN key: he will see only blinking icons between the admitted values (that are 1 and 3). The bit 9 will be set to 0
3. Application confirms speed: bit 9 = 1

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		1	0	1	0	1	1	1	0	1	1	0	0	0	0	0

To reset blinking set bit 8 to “0”

The “*BigArea*” 1 / 2

- The “BigArea” **shows** the value of **one** (fixed) or **more variables** (the *Carousel*)
- To define what to show use the **Cfg_big_area** variable



Cfg_big_area (dec. value)	Description	Variable involved
0	Displays the ***‘Carousel of variables’***: that is, only the variables specified in the **Bigarea_carosello_cfg** (however, it is possible to use **SmallArea** for display another variable, see restriction to SmallArea).	Bigarea_carosello_cfg
1	Displays the room temperature measured by the pCO in ° C or ° F.	C_temp
2	Displays the temperature setpoint in ° C or ° F	Set_Temp
3	Displays the room humidity measured by the pCO in %rH.	C_hum
4	Displays the humidity set point in %rH.	Set_Humid
5	Displays the external temperature measured by the pCO in ° C or ° F.	External_Temp
6	Displays the pressure measured by the pCO in PSI or BAR.	C_pressure
7	Displays value of ‘***free variable***’ n° 1 in BigArea, and acronym in SmallArea (overwriting what is configured in small area)	Free_vaule1_val Free_value_1_char12/34 Free_value_1_unit
8	Displays only value of ‘***free variable***’ n° 2 in BigArea, and acronym in SmallArea (overwriting what is configured in small area)	Free_vaule2_val Free_value_2_char12/34 Free_value_2_unit
9	Displays only value of ‘***free variable***’ n° 3 in BigArea, and acronym in SmallArea (overwriting what is configured in small area)	Free_vaule3_val Free_value_3_char12/34 Free_value_3_unit
10	Displays only value of ‘***free variable***’ n° 4 in BigArea, and acronym in SmallArea (overwriting what is configured in small area)	Free_vaule4_val Free_value_4_char12/34 Free_value_4_unit

Example: Set ***Cfg_big_area***=7 dec, in “BigArea” is displayed only “Free variable” n° 1

The “*BigArea*” 2 / 2

Example 1

• To show the room temperature read by internal sensor → Set **Cfg_big_area**=1 dec

• The “BigArea” also shows the “***Set_Temp***” value is being set by the user, when twisting the encoder (SET icon appears)

Example 2

• To show the room humidity read by internal sensor → Set **Cfg_big_area**=3 dec

• The “BigArea” also shows the “***Set_Humid***” value is being set by the user, when twisting the encoder (SET icon appears)

“Cfg_big_area” is not a bitfiled variable!

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						0	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**

The “*SmallArea*” 1 / 2

- The “SmallArea” **shows** the value of **a variable** or the **clock** (similar to “BigArea”)
- To define what to show use the **Cfg_small_area** variable

Cfg_small_area (dec. value)	Description	Variable involved
0	Displays the ‘Clock’ and shows the day of the week.	See clock variables
1	Displays the room temperature measured by the pCO in ° C or ° F.	C_temp
2	Displays the temperature set point in ° C or ° F	Set_Temp
3	Displays the room humidity measured by the pCO in %rH.	C_hum
4	Displays the humidity set point in %rH.	Set_Humid
5	Displays the external temperature measured by the pCO in ° C or ° F.	External_Temp
6	Displays only the VALUE of Free Variable 1 with unit its unit of measurement	Free_vaule_1_val Free_value_1_unit
7	Displays only ACRONYM of ‘***free variable***’ n° 1.	Free_value_1_char12/34
8	Displays only ACRONYM of ‘***free variable***’ n° 2.	Free_value_2_char12/34
9	Displays only ACRONYM of ‘***free variable***’ n° 3.	Free_value_3_char12/34
10	Displays only ACRONYM of ‘***free variable***’ n° 4.	Free_value_4_char12/34

- Some settings of **BigArea** override the ones of the **SmallArea:**

Set in BigArea	Displayed in SmallArea
C_temp	“tE”
Free_value*_val	Free_value_*_char12/34 (acronym of variable)
*	Can be 1, 2, 3 or 4

See also the chapter regarding the CUSTOM FUNCTION that can overwrite Small Area

The “*SmallArea*” 2 / 2

Example

- To show the room temperature read by internal sensor → Set **Cfg_small_area** =1 dec

“Cfg_small_area” is not a bitfiled variable!

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						0	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**

How to configure the "*Carousel*" 1 / 2

- "*Carousel*" is a *simple menu* for user with a short list of variables
- To select the variables included in the "*Carousel*" use **BigArea_carosello_cfg**

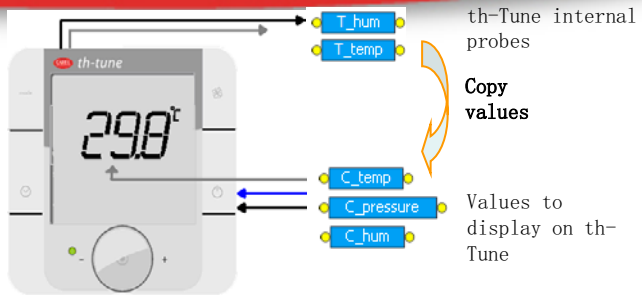
Bit	Description	Variable
0	Displays the room temperature measured by the pCO in ° C or ° F.	C_temp
1	Displays the temperature set point in ° C or ° F	Set_Temp
2	Displays the room humidity measured by the pCO in %rH.	C_hum
3	Displays the humidity set point in %rH.	Set_Humid
4	Displays the external temperature measured by the pCO in ° C or ° F.	External_Temp
5	Displays the pressure measured by the pCO in PSI or BAR.	C_pressure
6	Displays '***free value***' n° 1	Free_vaule1_val, Free_value_1_unit Free_value_1_char12/34
7	Displays '***free value***' n° 2	Free_vaule2_val, Free_value_2_unit Free_value_2_char12/34
8	Displays '***free value***' n° 3	Free_vaule3_val, Free_value_3_unit Free_value_3_char12/34
9	Displays '***free value***' n° 4	Free_vaule4_val, Free_value_4_unit Free_value_4_char12/34

- Browsing of the "*Carousel*"
  - Press the "**ENCODER**" to display next variable of "*Carousel".*
  - After 10s of inactivity is shown the first variable of the "*Carousel"*

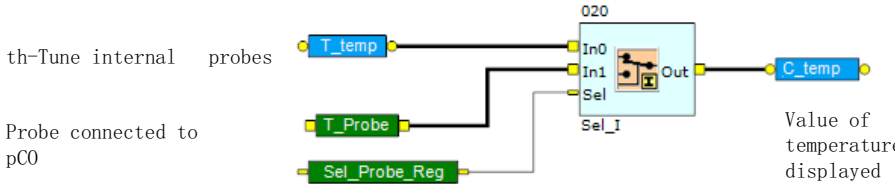
How to configure the "*Carousel*" 2 / 2

Example

- How configure **Bigarea_carosello_cfg** to display
  - Temperature and humidity read by internal probes of th-Tune
  - Pressure read by pCO probe



- Steps
  - Use this variables in *Strategy Editor*
    - Th-Tune internal probes
      - `T_temp, T_hum`
    - Value to display on th-Tune
      - `C_temp, C_hum, C_pressure`



  - Configure **Bigarea_carosello_cfg** = 37 dec

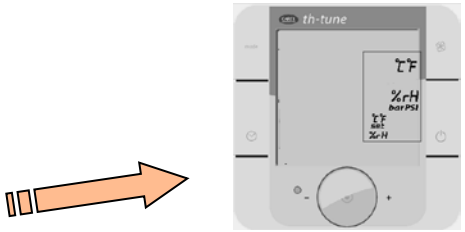
Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									0	**0**	**1**	**0**	**0**	**1**	**0**	**1**

  - Configure **Cfg_big_area** = 0 dec

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													0	**0**	**0**	**0**

Note

- It's <u>necessary to convert</u> only **Pressure** before displaying it
- The **Temperature** is <u>already converted</u> by terminal
- Use **Cfg_flags** variable
  - `Bit3:` to display ° C/° F symbol in Big/SmallArea
  - `Bit4:` to display BAR/PSI symbol only in BigArea



How to configure ”*Free variables*” 1 / 4

- The **Big / SmallArea** can show also two generic variables: the **Free variables**
- The **Free variables** must be **configured** before being used
- For each Free variables specify: value, 4-char acronym (i.e. “DIF1“), unit of meas, R/W properties and limits

this is an example with free variable 1

Variable	Description
Free_value_1_val	Specifies the value
Free_value_1_char12	Specifies the 1st and 2nd char (form right) (i.e. ‘1’ and ‘F’)
Free_value_1_char34	Specifies the 3rd and 4th char (form right) (i.e. ‘i’ and ‘d’)
Free_value_1_unit	Specifies the unit of measurement / decimal point

Note

- Selecting **BigArea** to display the free variable vaule, in **SmallArea** will be displayed acronym of free variable
- Selecting **SmallArea** to display the free variable vaule, in **SmallArea** will be displayed **only** acronym of free variable and not variable value.

How to configure ”*Free variables*” 2 / 4

Acronyms

Example

Display in **BigArea** variable “Differential” using 1st free variable

Steps

- Configure **Free_value_1_char12 / 34** to display in **SmallArea** ‘DIF1’
  - Char: ‘A’ = 65dec = 0100 0001bin, etc
  - In **SmallArea** string ‘DIF1’ (‘diF1’ displayed):

    ‘D’ = 68 dec = 01000100 bin (use CHARMAP/CALC to find char code)
    ‘I’ = 73 dec = 01001001 bin
    ‘F’ = 70 dec = 01000110 bin
    ‘1’ = 49 dec = 00110001 bin

  - **Free_value_1_char12** =17969 dec

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1

2nd char: ‘F’ 1st char: ‘1’

  - **Free_value_1_char34** =17481 dec

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	1

4th char: ‘D’ 3rd char: ‘I’

- Value displayed in **BigArea**
  - Configure **Free_value_1_val** = 90 dec

How to configure ”*Free variables*” 3 / 4

Variable configuration

- Value displayed in **BigArea**

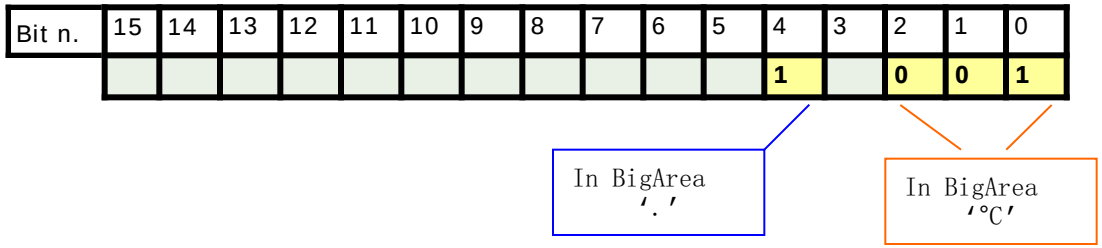
Configure **Free_value_1_unit** to set
- unit of measurement
- decimal point (all the exchanged variables are integer)
- R, R/W property
- Maximum and minimum value that can be set by the user with the encoder for R/W variables

Bit	Description
0-2	Unif Of Measurement 000: (none) 001: °C 010: °F 011: %rH 100: BAR 101: PSI 111: ppm (for custom thTune)
3	If the bit is “0” the variable can be written by the user with the encoder (set to 1 for read only variables)
4	Hide decimal point 0: the ‘.’ is shown (in the **BigArea**) 1: the ‘.’ is not shown (in the **BigArea**)
5,6	Multiplier for minimum & maximum 00: x1 01: x10 11: x100 11: x1000
7-10	Maximum value (only positive) from: *0* to *15* (=1111)
11-15	Minimum value (also negative) from: *-15* (=10001) to *15* (=01111)

How to configure ”*Free variables*” 4 / 4

Examples

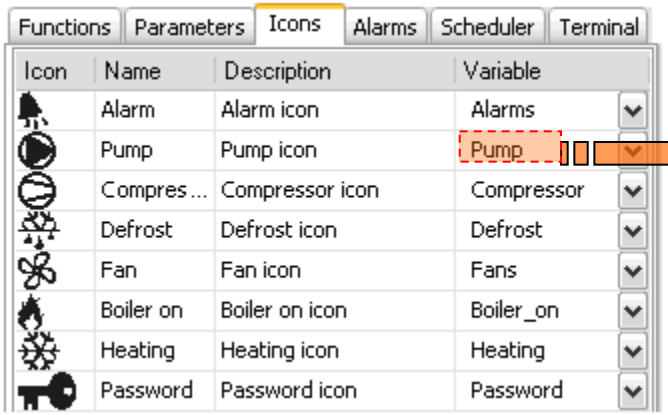
- Value displayed in **BigArea**
  - Configure **Free_value_1_unit** to set unit of measurement and decimal point
  - Set **Free_value_1_unit =** 17 dec
  - Set **Cfg_big_area =** 7 dec



Icons

• It is possible to manage the icons in the lower part of the display through some variables

•When using th-Tune editor the variables in the **Icon** tab define the status of icons



Status of icons

Value	Description
0	Symbol OFF
1	Symbol ON
2	Symbol blinking

Example: Blinking  ➔ **Defrost** = 2

• When connecting the th-Tune in Modbus master it's necessary to write correspondent holding registers to those values (0-1-2) to obtain the same result

•**Note for ALARM icon**

- Application can switch on symbol  ,
- If an alarm is present, application doesn't switch off  symbol, using "Alarms" variable.

Management of keys pressed on the terminal 1 / 3

Key_buffer

- contains the *code* of key, or pair of keys pressed
  - (Example :“**FAN**” or ”**MODE+FAN**”)

- The code is automatically set when pressing:
  - A single key
  - A couples of keys

- The *code* of key pressed and the “*pressure time*”
  - are maintained in **key_buffer** and **key_buffer** till a new key is pressed

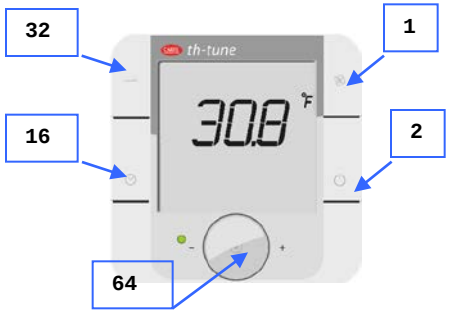
Management of keys pressed on the terminal 2 / 3

- **Key_buffer**
  - It's used as bitfield variable
  - It contains the code of key pressed (see table)
  - In **Key_timer** is inserted the "*pressure time*" of the key pressed
  - The code of key pressed and "*pressure time*" are maintained in **key_buffer** and **key_buffer** till a new key is pressed

Bit	Description
0	Key "FAN": 1 pressed
1	Key "POWER ON/OFF": 1 pressed
2	Not used for this type of terminal
3	Not usedfor this type of terminal
4	Key "CLOCK": 1 pressed
5	Key "MODE": 1 pressed
6	Key "ENCODER": 1 pressed
other	Reserved

Example1

- A single key pressed
  - FAN (BIT0=1 → 1dec)



Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									0	0	0	0	0	0	0	1

Example2

- A couples of keys pressed
  - POWER_ON/OFF & CLOCK (BIT1=1 and BIT4=1 → 18dec)

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									0	0	0	1	0	0	1	0

Management of keys pressed on the terminal 3 / 3

Key_timer

- In the first part of **Key_timer** (bit 0-7) is inserted the "*pressure time*" of a key:
  - 0: key is pressed and immediately released
  - 1..16: The key has been pressed for 1, 2 or 16 seconds since last read
  - keys sampling every ~250ms.

- Description of behavior of **Key_Timer**:
  - When the key is pressed, Key_Timer is immediately set to "**0**".
  - Meanwhile, if the button is pressed firmly for at least one second, Key_Timer will get an increment value starting from "**1**" and increasing every ~250ms, reaching a top value of "**16**" (corresponding to 16 seconds). When 16 is reached, it will not be incremented anymore.

- The bits 8-15… represent a counter of how many times a specific key is pressed until a new one is pressed updating key_buffer

Example

•FAN key was pressed for 3 times and now is beeing pressed since 7 seconds

» Key_buffer

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									0	**0**	**0**	**0**	**0**	**0**	**0**	**1**

» Key timer

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**

Key_enable_mask

- It is used to bitfield to *enable* or *not*, the functions associated with keys on the physical keyboard,
- Some bits are used for a “Custom Function”

Bit	Description
0	Editing “FAN” key by phisical keyboard: 1 disable, 0 enable
1	Editing “MODE” key by phisical keyboard: 1 disable, 0 enable
2	Editing “*Clock Hour*” (HH,MM,Day) pressing “CLOCK” key: : 1 disable, 0 enable
3	Editing “*Time Bande*” pressing “CLOCK” key: 1 disable, 0 enable
4	Editing “*Temperature Setpoint*” using “ENCODER” key: 1 disable, 0 enable
5	Editing “*Humidity Setpoint*” using “ENCODER” key: 1 disable, 0 enable
6	Showing “Alarms loop” pressing “MODE+CLOCK” keys: 1 disable, 0 enable
7	Showing “*Res Alr*” property in “*Alarms loop*”: 1 disable, 0 enable
8	Disable “CLOCK” key short-press to enable/disable “*Time Bands*”: Bit8= 0 “*short-press*” enable, Bit8= 1 “*short-press*” disable (see slide 12). NOTE: BIT1 of “*Status_flags*” variable can be used by application to display incon
9	Showing machine parameters pressing “FAN+POWER ON/OFF”
10	Enable custom function with up right key (“POWER ON/OFF”)
11	Enable custom function with middle right key (“FAN”)
12	Enable custom function with up left key (“CLOCK”)
13	Enable custom function with middle left key (“MODE”)
14	Visualize the Acronym of Free variable 3 in small area (Overrides the Cfg_Small_Area setting)
15	Visualize the Value of Free variable 3 in small area (Overrides the Cfg_Small_Area setting)

Keys functionality lock

Setting the bits in Key_enable _mask

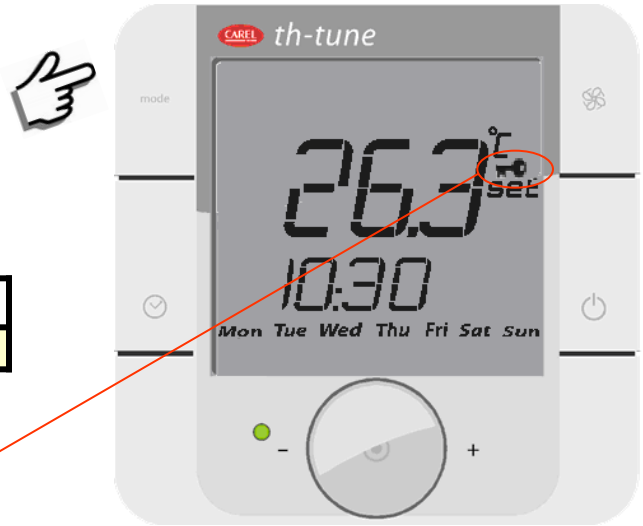
- Only the functions associated with keys on the physical keyboard are locked it's possible to continue to read if a key is pressed and to enter other functions associated with that key
- "**POWER ON / OFF**" key cannot be locked
- The default value is all function keys enabled (all bit to 0)
- When trying to access a locked function the icon "KEY" appears for one second

Example

- Disable function keys
  - "**FAN**"
  - "**MODE**"

Settings in Key_enable_mask

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**



Pressing MODE the "KEY" icon appears showing that the function is locked

“Custom Function”

- It is possible to configure one or more buttons to enable a “Custom Function”
- The activation of the Custom Function will set the bit 4 of “ *Status_flags*” variable at “1” for using this information inside the application
- it is possible to modify the visualization of the main screen overwriting the *small area* or the *big area* with the values of free variable 3 (see section..)

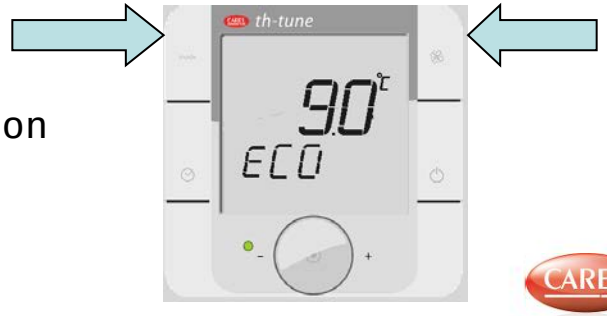
Bit	Description
10	Enable custom function with up right key (FAN)
11	Enable custom function with middle right key (ON/OFF)
12	Enable custom function with up left key (CLOCK)
13	Enable custom function with middle left key(MODE)
14	Visualize the Acronym of Free variable 3 in small area
15	Visualize the Value of Free variable 3 in small area

Settings on “*Key_enable_mask*” variable

Overrides the Cfg_Small_Area setting

Example: pressing “*FAN*” and “*MODE*” to enable “ECO” function

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**



Management of “CLOCK” 1 / 2

Clock Key

“**CLOCK**” key

- Short press of “**CLOCK**” key
  - Enable/disable “*Time bands*”
    - In “***Status_flags”*** variable → Bit1 change state
    - symbol is displayed
    - As a default the bit

Time bands enabled

Time bands disabed

- Press “**CLOCK**” key for 2s to enter in “*Clock menu*”
  - 3 selections are possible and then push “**ENCODER**” to confirm a selection:
    - “*CLOC*”
    - “*TIME BAND*”
    - “*ESC*” (ON-OFF button as a shortcut)

Selecting “***CLOC***”

- To configure in sequence: “*Hour”*, “*Minute”* and “*Day”*
- Rotate “**ENCODER**” to change value and push to confirm
- The variables can be changed through the serial line

NOTE

It is not possible to enter in clock setting mode, if time bands are enables (**Clock** icon ON)

Management of "CLOCK" 2 / 2

Clock Visualization and variables

- Terminal can display in SmallArea
  - Hours of the th-Tune internal clock
    - Application can only read it on these variables:
      - T_clock_hrs, T_clock_min, T_clock_day
      - Pressing "CLOCK" key for 2s is possible to change them

  - Hours sent by pCO
    - Application can write/read it on these variables:
      - C_clock_hrs, C_clock_min, C_clock_day
      - Pressing for "CLOCK" key for 2s is possible to change them, and when using th-Tune protocol, BIOS change automatically *system variables* (*CURRENT_HOUR*, *CURRENT_MINUTE*)

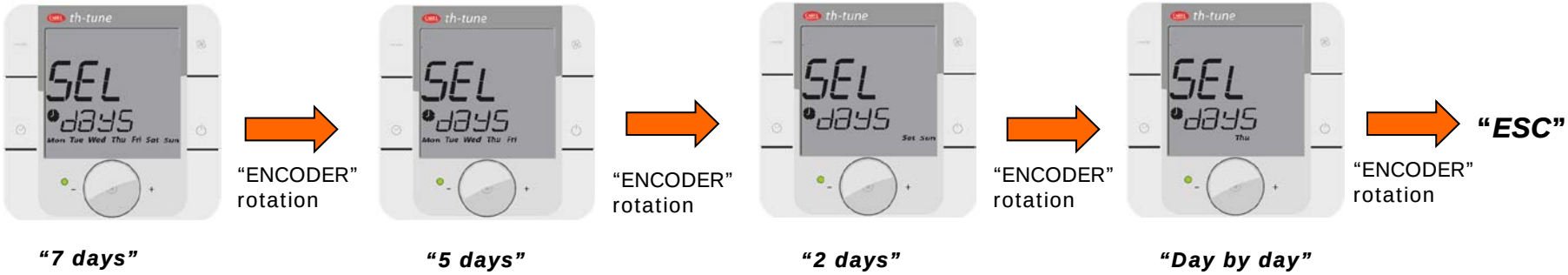
- Use Bit9 of **cfg_flags** variable to set hours to display
  - Bit9=0 display internal clock (th-Tune display: T_clock_hrs, T_clock_min, T_clock_day)
  - Bit9=1 display pCO clock (th-Tune display: C_clock_hrs, C_clock_min, C_clock_day)

Time Bands Management 1 / 5

Time Bands Selection

Selecting "***TIME BAND***" after pushing CLOCK button for 2 s

- After the push of "**ENCODER**" terminal displays "*Sel days*"
  - Rotating "**ENCODER**" is possible to select a group of days or a single day:
    - ***"7 days"*** (mon, tue, wed, thu, fri, sat, sun)
    - ***"5 days"*** (mon, tue, wed, thu, fri)
    - ***"2 days"*** (sat, sun)
    - ***"Day by day"***
    - ***"ESC" ("ON-OFF" is a shortcut for this option)***



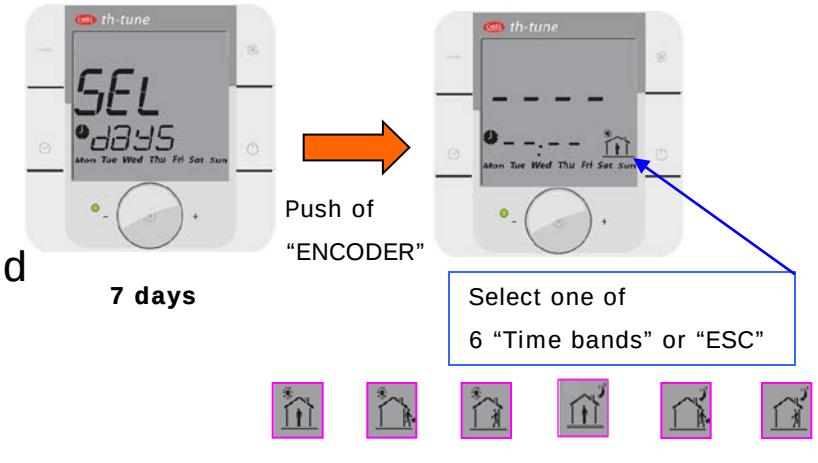
Example

- Push on "**ENCODER**" to confirm a selection (i.e. "*7 days*")
  - Then terminal asks to the user to select **one** of the "**Active Time bands**"

Time Bands Management 2 / 5

Time Bands Setting

- How to select single “**Time band**”
  - From “*Sel day*” select “*7 days*”
  - Terminal asks to select:
    - One of the 6 “Time bands”
    - *“ESC” (this option can be*



- Parameters of a single “**Time band**” selected
  - “***Temperature Setpoint***”
  - “***Start time***”

- How to modify parameters
  - Push “**ENCODER**”
  - In the sequence is possible to change
    - “*Hour*”, “*Minute*”, “*Setpoint*”
    - Push “**ENCODER**” to confirm each parameter

- Notes

The sequence of time bands is fixed and it is possible to view and end it only following this sequence (i.e. to go back it’s necessary to keep rotating the encoder to start again from the beginning.

Remember that at the application level it’s possible to disable and make disappear a time band using “*cfg_flags*” variable.

Time Bands Management 3 / 5

Time Bands Setting

- How to "***Disable a band***"
  - In "*Hour*" field rotate "**ENCODER**" up to display "--:--"
  - Push "**ENCODER**" to confirm

Note

- Use these value (24, 60 and -10000) to disable a band by application, and display on terminal "--:--"
- Remember that at the application level it's possible to disable and make disappear a time band using "*cfg_flags*" variable.

- How to display the state "***Off machine***"
  - In "*Setpoint*" field rotate "**ENCODER**" up to display "*OFF*"
  - Push "**ENCODER**" to confirm

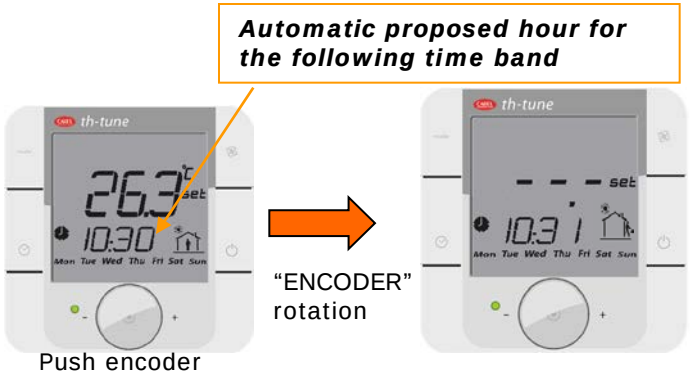
Note

- At the application level it's possible to read in the setpoint variable, this value:
  - "*Setpoint*"= -10000
- When using the internal scheduler of th-Tune also ON OFF variable is set

Time Bands Management 3 / 5

Overlapping control

- The data are stored in the th-Tune so when passing from a time band (hh:mm) to another one the display is proposing as starting time the hour and minute following the previous one (hh:mm+1)
- The th-Tune is checking the consistency of the time bands
  - When editing the time band: if the exhisting hour is overlapping the previous time band a compatible value is proposed
  - When trying to escape from time bands setting (ESC) with a global check: if there is any overlapping it is not possible to go out of the menu and the th-Tune goes directly to the first “inconsistent” time band proposing a compatible hour



NOTE: when changing a single day on a pre existent “multi – day” time bands, if going back in this multi day time band the inconsistent time band is disabled.

Time Bands Management 4/5

Time Bands Actions

When the time bands are enabled

At the HH:MM of the time band

- Setpoint variable is set to the stored value
- Current timeband is updated (icon is changed)
- If the time band is set to OFF the on off variable (**POWER**) is updated (=2 OFF from timeband

- When time bands are enabled key lock functionality is automatically set to disable set editing, but it's always possible to unlock the function using "*key_enable_mask*" variable.

Time Bands Management 5/5

Time Bands settings by BMS in Modbus only from 1.3 firmware release

- It is possible in Modbus to read and write the values for time bands used by th-Tune (registers from 256 to 339)
- The format of these information is the following "#Day Event Time #"

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Not used		Day of week (from 0 to 6)			Hour (from 0 to 23)					Minute (from 0 to 59)					

- Another variable "#Day Event Setpoint #" is the setpoint
- To load those variables into th-Tune it's necessary to set "config_flag_5" to 1 (automaticaly set to 0 by th-Tune)

NOTE

- We suggest to disable time band editing while updating the info to keep consistency
- Writing all variables can be a long procedure so an idea is to set the terminal in "Init" mode using "*config_flag_0*"

Internal alarms of the th-Tune

Alarms_Mask1

- This variable collects all the alarms of the devices connected to th-Tune terminal:
  - Internal temperature probe of th-Tune
  - Board with temperature probe
  - Board with humidity probe
  - Board clock
- It's used as bitfield (see table)

Bit	Description
0	Alarm on the internal temperature probe of th-Tune
1	Alarm on second temperature probe in the t+h board (t+h models)
2	Alarm on humidity probe in the t+h board
3	Alarm on the temperature probes (it's = BIT0 OR BIT1)
4	Alarm on the board clock
other	Reserved

Example

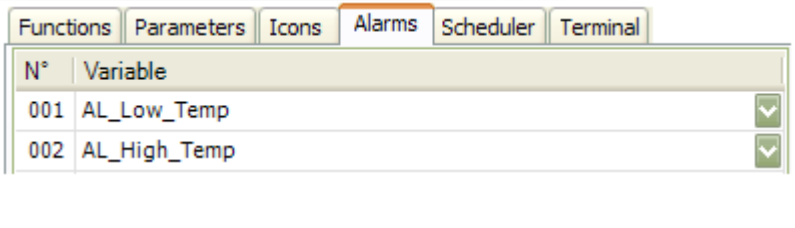
- **Alarm on**
  - Board with humidity probe
  - Board clock

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												1	**0**	**1**	**0**	**0**

“Alarms” tab 1 / 3

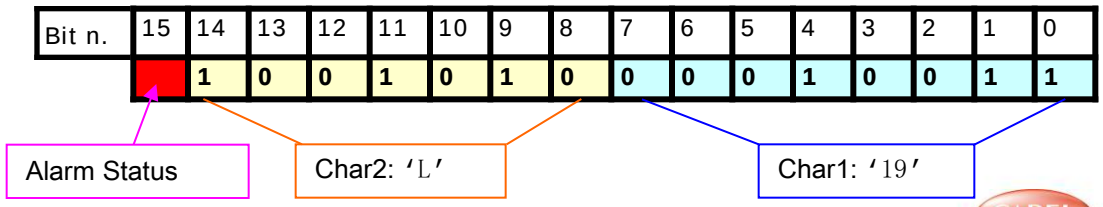
Inside ***Alarms*** tab :

- You can add some “*Alarm Variable*”
  - Each variable is *Integer*
- For each “*Alarm variable*” you can define:
  - *‘Alarm status’*
    - Bit15=0 alarm OFF
    - Bit15=1 alarm ON
  - *‘Alarm code’* is composed from 2 chars (i.e. “L19”)
    - **Char2** is one single alphabetic character:
      - use ASCII table(“A”=65dec,“B”=66dec,…)
      - use Bit8-14 = “L”= 1001010bin
    - **Char1** is a number:
      - range 0..99
      - use Bit0-7 = “19”= 00010011bin
  - Char ‘A’ is put by terminal



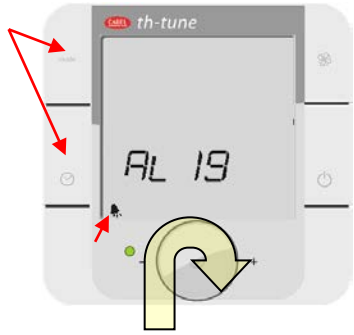
Order of displaying on th-Tune “***Alarms Loop***”

- How to configure “**AL_Low_Temp**” alarm variable:
  - Bit15: 0 alarm OFF, 1 alarm ON
  - Bit8-14 = “L”= 1001010bin
  - Bit0-7 = “19”= 00010011bin



“Alarms” tab 2 / 3

- **Diplaying “*Alarms Loop*”**
  - When an alarm is active “**Alarm icon**” is blinking
  - Press “**MODE+CLOCK ON/OFF**” keys for 3s to enter in “Alarms Loop”
  - rotate “**ENCODER**” to browse all active alarms
  - Press “**Res Alr**” to try to reset all active alarms
    - This is <u>only a request</u> by terminal to the application
    - Application has charge
      - to reset all “***Alarm variables***” → for each variable set the Bit15=0 (“Alarm Status” bit)
      - to manage “**Alarm_Flag**” variable
  - Press “**Esc**” to exit from the “**Alarms loop**” without reset no alarms. Pressing the ON-OFF button emulates the selection of this option



- How to reset “**AL_Low_Temp**” alarm variable

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1

Alarm Status

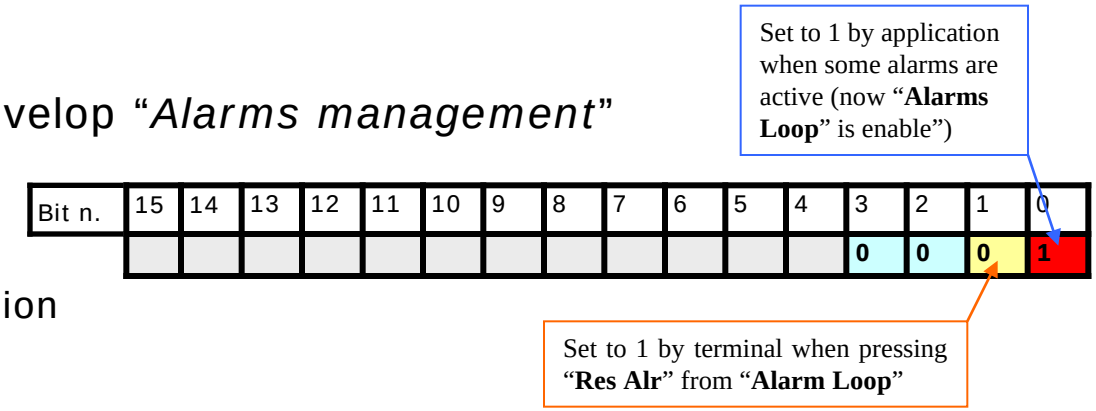
Char2: ‘L’

Char1: ‘19’

“Alarms” tab 3 / 3

Alarm_Flag variable

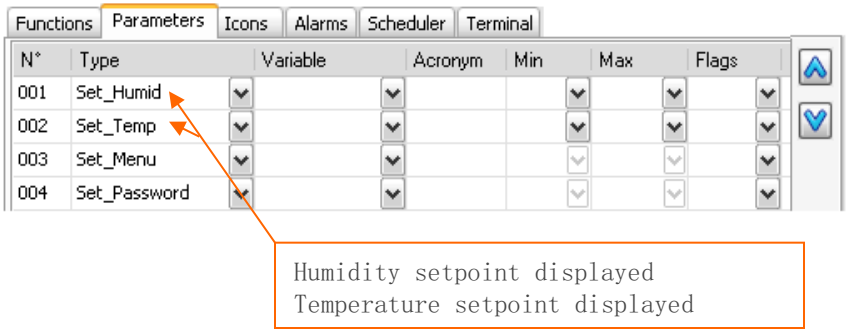
- Use **Alarm_Flag** variable to develop “*Alarms management*”
  - Bit0: Generic alarm
  - Bit1: Reset buzzer by user
  - Bit2: Buzzer status by application
  - Bit3: Buzzer mute



Bit	Name	Description
0	Generic Alarm	How to manage from application “*Generic Alarm*” bit: • The application sets the Bit0=1 and Bit2=1 when an alarm condition is active: – In your application Bit0 is the **OR** of all active alarms in the system. • Immediatly terminal switches on symbol and makes it blink How to have access to the “*Alarms Loop*” • Press “MODE+CLOCK” for 3s to enter in “*Alarms Loop*” and use “ENCODER” to display active alarms. • Select “*Res Alr*” to reset all active alarms, this selection sets Bit1=1.
1	Reset request by user	When selecting “*Res Alr*” inside “*Alarms Loop*” → Bit1=1 (it’s a request to the application). • When Bit1=1 the application recieves the request from terminal to try to reset all active alarms (Bit0) • Application has to reset Bit0, Bit1 and Bit2 (Buzzer sate is managed by application with Bit2).
2	Enable buzzer	If Bit3=0 then application can menage buzzer state: Bit2=0 switch off buzzer, Bit2=1 switch on buzzer
3	Buzzer mute	Buzzer is always switched off until Bit3=1
other	Reserved	Not used

“Parameters” tab 1 / 8

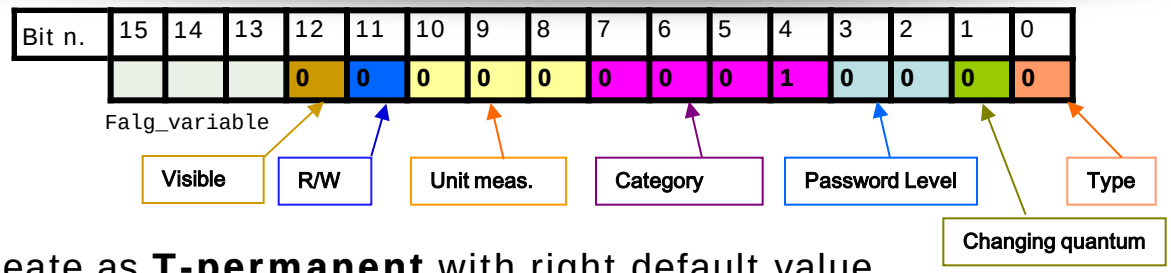
- The **Parameter** tab allows to define the “Parameters”
  - You must drag&drop variables from **Variable List**
- For each *parameter* you can define:
  - **Type**:
    - Set_Temp
    - Set_Humid
    - Set_Password
    - Set_Menu
    - (None)→ Define a Parameter
  - **Variable**
  - **Acronym** (i.e. “P12”)
  - **Min / Max**
  - **Flags**



- **Limitations**
  - Max 1 “**Set_Temp**” and “**Set_Humid**”
    - Acronym and Flags are ignored, because are dispalyed using ‘***Carousel***’
  - Max 4 variables type “**Set_Password**”
    - do not use ‘22’ or ‘0’ as password value
  - Max **15** variables type “**Set_Menu**” (1-15)
  - Max 4 chars for **Acronym**

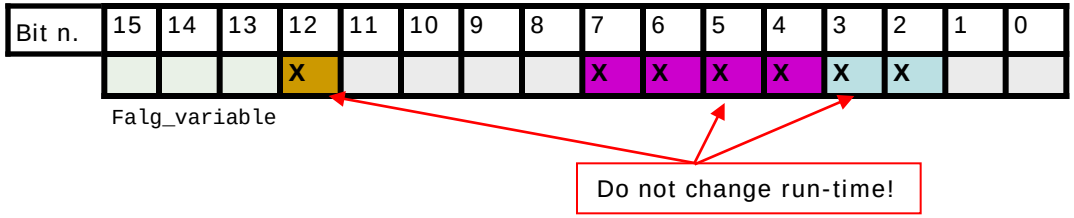
“Parameters” tab 2 / 8

- Structure of **Flag** variable



Limitations

- All Flag variables must be create as **T-permanent** with right default value
- **DEV** file must be upload inside controller
- **Do not change** at run-time for **all** Flag variables:
  - Bits value for ‘***Visibile***’, ‘***Category***’ and ‘***Password Level***’
  - these bits are necessary to th-Tune
    - To create association between ‘***Category’***, ‘***Parameter’*** and ‘***Password Level***’
    - To create the right file for BIOS to manage th-Tune



NOTES

- It’s not possible to set Acronym/Flag properties for “**Set_Temp**” and “**Set_Humid**” variables
  - Flag variables are **zero**, you can manage these variables only with ‘***Carousel***’
  - To place these variables in a *Category* with password see appendix
- It’s not possible to set “0000” as a password
- If one parameter is “read only” then it is automatically refreshed each 2 seconds during visualization
- First category is 1 :**4-7 bits vary from 0001 to 1111**. 0000 gives an error when compiling

“Parameters” tab 3 / 8

This table lists properties of **Flag** variable

Bit	Name	Description
0	Type	Number type 0 = Integer number 1 = Decimal number (put decimal point on display)
1	Changing Quantum	0 = Parameter changes with step 5 (20.5 →21.0) 1 = Parameter changes with step 1 (20.5 →20.6)
2-3	Password Level	At each parameter is necessary to associate a Password Level. 00 = Level 0 (password with Level 0) ← lower access level 01 = Level 1 (password with Level 1) 10 = Level 2 (password with Level 2) 11 = Level 3 (password with Level 3) ← higher access level Press ‘FAN+Power ON/OFF’ for 3 sec, then terminal asks to insert password to start browsing inside Categories and their parameters. Within a category are not visible parameters with password level greater than password level specified. When specifing password with ‘Level 3’ (higher access level), it is possible to see every parameter in all Categories.
4-7	Category	Specifies which Category (max 15) the parameter belongs. It is possible to associate a password for each parameter inside a Category. The visibility of the parameters is dependent on the level of the password. 0001 = Category 1 … 1111 = Category 15
8-10	Unit measurement	000 = Pure number 001 = Temperature 010 = Humidity 011 = Pressure
11	R/W	Read or Write the parameter 0 = Read 1 = Read/Write
12	Visible	When this bit is set it allows to hide a parameter inside a Category 0 = Visible 1 = Not visible
other	Resrved	Not used

“Parameters” tab 4 / 8

Example

- To create two **Categories** everyone with a **Password** and three **Parameters**
  - In this example I suggest you a rapid way to create association between ***Menu***, ***Password*** and ***Parameters***
  - I use the same ***Key*** (16 and 36) to identify quickly the association on the “***Parameters tab***”
  - Then I use “***Strategy Editor***”to configure correctly each single *parameters.*

Bit n.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				1	1	0	0	0	0	0	1	0	0	1	0	1

Parameters

Do not change **Bit2-7** and **Bit12** with *Strategy Editor*.

- NOTE
  - Some bits are necessary to create the *Category* to display when pressing “**FAN+POWER ON/OFF**” keys
  - For each *parameter* is not possible to change *Category, Password level,…* (BIT 2-7/12) at “*run-time*”
  - During compilation of a solution 1tool associates each *parameter* to a single *menu* in permanent mode
  - To change *category* to a *parameter* is necessary to modify solution, and then upload Binary + DEV files

Functions | Parameters | Icons | Alarms | Scheduler | Terminal

N°	Type	Variable	Acronym	Min	Max	Flags
001	Set_Humid	Th_Tune_Set_H		Th_Tune_Set_H_min	Th_Tune_Set_H_Max	thTune_Term1_Humid_Flag
002	Set_Temp	Th_Tune_Set_T		Th_Tune_Set_T_Min	Th_Tune_Set_T_Max	thTune_Term1_Temperature_Flag
003	Set_Menu	thTune_MNU1_MENU	MNU1			Menu1_Flag = 16
004	Set_Menu	thTune_MNU2_MENU	MNU2			Menu2_Flag = 36
005	Set_Password	thTune_PSW_MNU1 = 12		Th_tune_PSW_Min	Th_tune_PSW_Max	PSW1_Flag = 16
006	Set_Password	thTune_PSW_MNU2 = 34		Th_tune_PSW_Min	Th_tune_PSW_Max	PSW2_Flag = 36
007		var_MNU1_1	A001	var_MNU1_1_min	var_MNU1_1_Max	var_MNU1_Flg_1 = 16
008		var_MNU1_2	A002	var_MNU1_2_min	var_MNU1_2_Max	var_MNU1_Flg_2 = 16
009		var_MNU1_3	A003	var_MNU1_3_min	var_MNU1_3_Max	var_MNU1_Flg_3 = 16
010		var_MNU2_1	B001	var_MNU2_1_min	var_MNU2_1_Max	var_MNU2_Flg_1 = 36
011		var_MNU2_2	B002	var_MNU2_2_min	var_MNU2_2_Max	var_MNU2_Flg_2 = 36
012		var_MNU2_3	B003	var_MNU2_3_min	var_MNU2_3_Max	var_MNU2_Flg_3 = 36

Set = 0

Password

Key

Parameters of Menu_2

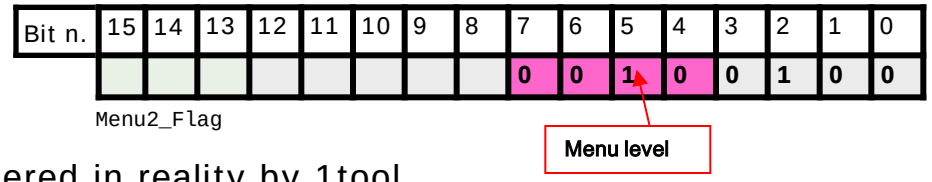
“Parameters” tab 5 / 8

An other way

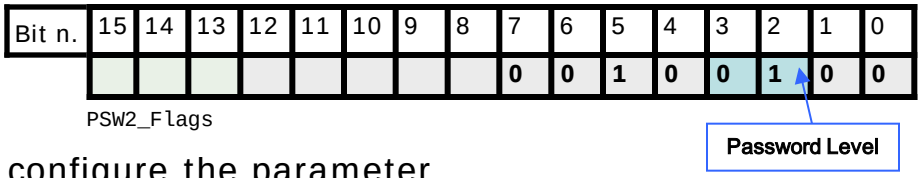
- It’s to set only the right bit for each type (*Menu, Password* and *Parameter)*
- But it’s difficult to read in “***Parameters tab***”

Example

- For **Set_Menu** types only Bit4-7 are considered in reality by 1tool
  - Menu2_Flag = 36 dec



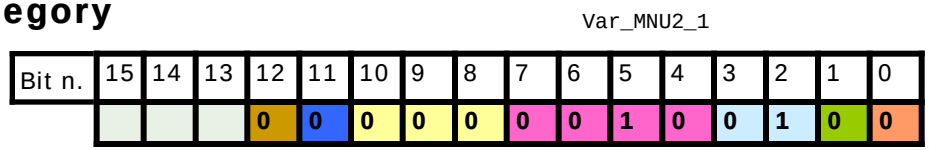
- For **Set_Password** types all Bit2-3 are considered in reality by 1tool
  - PSW2_Flags = 36 dec



- For **Parameter** types only Bit0-12 are used to configure the parameter
  - Do not change “run-time” Bit4-7
  - Is possible to hide a parameter inside a ***Menu*** giving it a password level higher (Note: 4 level of password)
  - Here the parameter is Read only so, during visualization it will be refreshed real time even without touching the encoder (that does not happen with R/W parameters). After 90 seconds the th-Tune goes back to main menu (10 s with backlight, 70 s no backlight, 10 s blinking)

How to display **var_MNU2_1** parameter in a **Category**

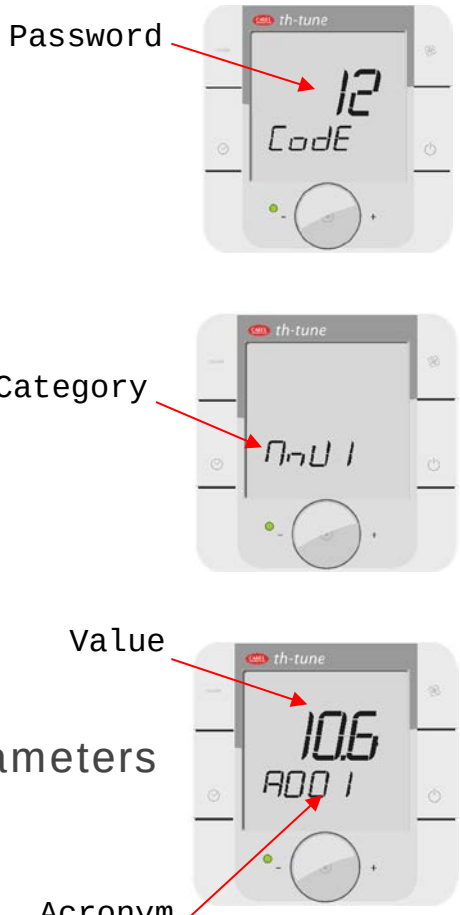
- var_MNU2_Flg1 = 36 dec
- At run-time from Strategy is possible
  - use other Bits to configure the parameter



“Parameters” tab 6 / 8

Example

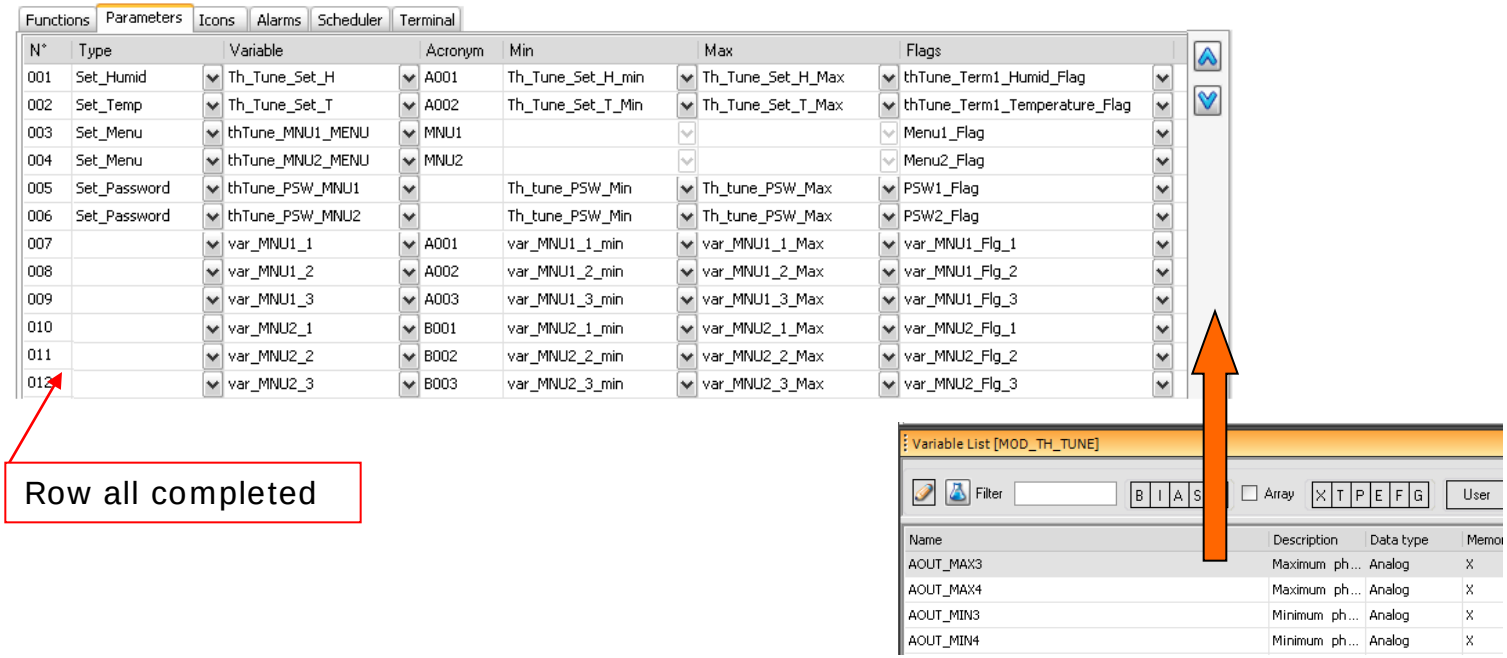
1. Upload software inside pCO
2. Press “FAN+Power ON/OFF” for 3s on th-Tune
3. Insert password “12” and press wheel to confirm
4. Now you can browse categories rotating wheel
   1. MNU1→ MNU2 → ESC → …
   2. use “ESC” to exit from editing parameters
5. Select “MNU1” and press wheel to confirm
   1. Now you can browse parameters rotating wheel
   2. A001→ A002 → A003 → ESC → … (OR on OFF button)
   3. use “ESC” to exit from “MNU1”
6. How to change a parameter
   1. Select “A001” parameter press wheel
   2. rotating wheel you can change “A001” parameter value
   3. press wheel to confirm the new value
7. If you select “MNU2” with password “12” you see no parameters
   1. Use “ESC” to exit from editing parameters
   2. Press “FAN+Poewr ON/OFF” for 3s on th-Tune
   3. Insert password “34” and press wheel to confirm
   4. Now you can see all parameters in each categories



“Parameters” tab 7 / 8

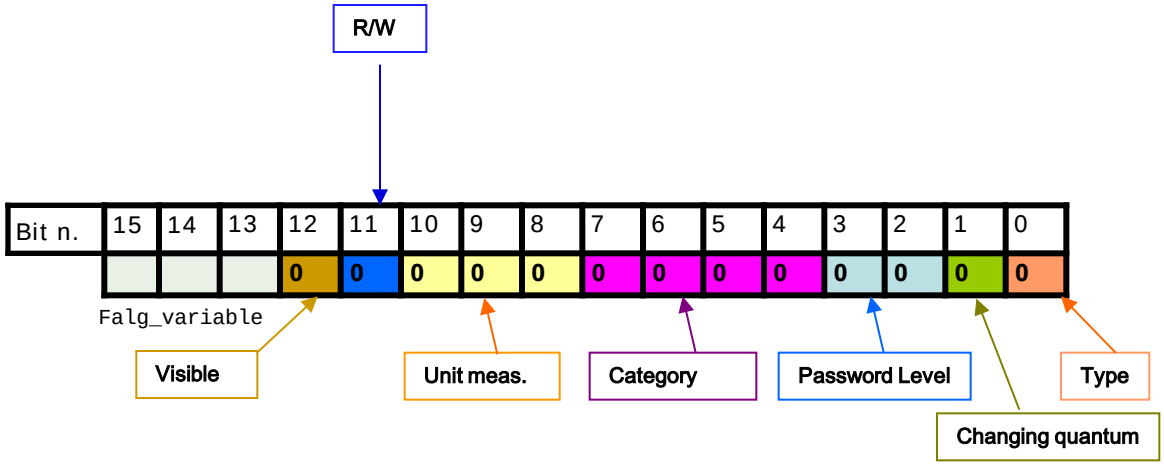
How add variables when row are all completed

- Darg&Dropp variable from **Variable List** where indicate by arrow
- So you can add a new row with the variable inside



Parameters refresh 8 / 8

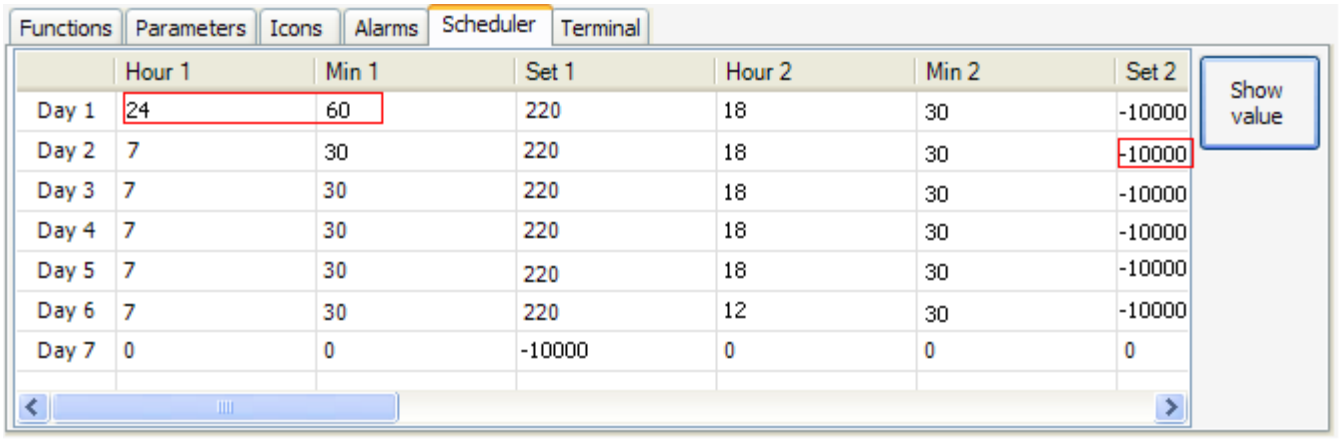
- Now setting the “read only” bit on parameters property this variable is automatically refreshed even without any operation made by the user: this is useful for monitoring real time changes of variables like probe reading
- The behaviour of the visualization is different: 10 seconds of backlight, then 70 seconds of visualization then blinking for 10 seconds before going out of the menu
- In parameters menu it’s available the new funtion of ON-OFF button that is doing the same as “ESC” options in the menu



“Scheduler” tab NOT USED

This tab collects the data of the scheduler sent by the 1.0 firmware version that did not manage internal scheduler.

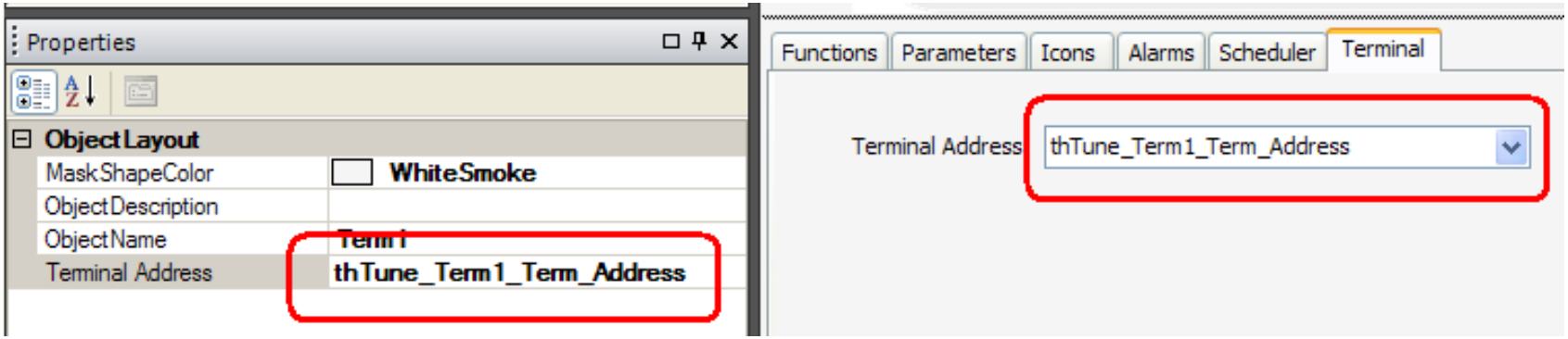
The feedback we received on usage difficulty of this method led us to change the way of manage the scheduler: as a consequence this tab



	Hour 1	Min 1	Set 1	Hour 2	Min 2	Set 2
Day 1	24	60	220	18	30	-10000
Day 2	7	30	220	18	30	-10000
Day 3	7	30	220	18	30	-10000
Day 4	7	30	220	18	30	-10000
Day 5	7	30	220	18	30	-10000
Day 6	7	30	220	12	30	-10000
Day 7	0	0	-10000	0	0	0

“Terminal” tab NOT USED

- The address of the th-Tune terminal can be set in the **Terminal** tab or in the ***Properties.***
- the fact that it is possible to connect just one terminal makes this feature unuseful; it’s enough just to keep address 1 otherwise it’s necessary to set the right address on the th-Tune



Terminal parameters

How to access Terminal parameters

- Press '**FAN'** e '**Power ON/OFF'** for 3s
- Enter password '**22**'

NOTE

- Do not use value '22' or '0' as password for **Categories** otherwise you have access terminal parameters and not to Categories

Acronym	Description	Default Value
Addr	Serial Address	1
bAud	Baud rate	2
bLbE	Backlight behaviour	0
bLIn	Backlight intensity	2
PCaL	Probe calibration	
CnSt	Contrast	15
P_In	Bypass Init if >0	0
PSuI		22
Year		1
Mont		1
nday		1
uday		1
hour		0
Mins		0
ESC		-