



# MCC

## Multiroom Climate Controller

ZCL-MCC

Application program version: [1.2]  
User manual edition: [1.2]\_a

# CONTENTS

---

Contents .....	2
Document Updates .....	3
1 Introduction .....	4
1.1 MCC (Multiroom Climate Controller).....	4
1.2 Installation.....	5
2 Functionality.....	7
2.1 Configuration Check.....	7
2.2 Operation Roles.....	9
2.2.1 Master .....	9
2.2.2 Slave .....	11
2.3 Master/Slave Role Selection .....	14
2.4 LED Notifications .....	15
2.5 Initial States.....	16
3 Configuration.....	18
3.1 General.....	18
3.2 Hospitality Thermostat.....	20
ANNEX I. Communication Objects.....	21

## DOCUMENT UPDATES

---

Version	Changes	Page(s)
[1.2]_b	<b>Changes in the application program:</b> <ul style="list-style-type: none"><li>• Update of the Hospitality Thermostat function to version 0.2.4 (corresponding user manual: 0.2_b).</li></ul>	-

# 1 INTRODUCTION

---

## 1.1 MCC (MULTIROOM CLIMATE CONTROLLER)

---

**MCC** (Multiroom Climate Controller) from Zennio is an in-line controller aimed at managing and controlling the climate of multiple hotel rooms (or any other type of guest rooms, e.g., residential or hospital rooms). To that end, it is possible to enable **up to fourteen thermostats independently**.

Given how critical it is that a single device controls the climate of several rooms, it is possible to have two exactly alike MCCs (and with the same configuration) working in the same system, **one of them as a master and the other as a slave**. Thus, the master device will operate normally while the slave remains on standby to take control if the master fails.

The most outstanding features of MCC are:

- System configuration for operation with a **single device** or two devices with **Master-Slave** roles.
- Up to **14 independent Hospitality thermostats**.
- Communication objects and LEDs indicators of **master/slave role, slave error** and **configuration error**.

## 1.2 INSTALLATION

MCC connects to the KNX bus through the on-board KNX connector. Once the device is provided with power from the KNX bus, both the individual address and the associated application program can be downloaded.

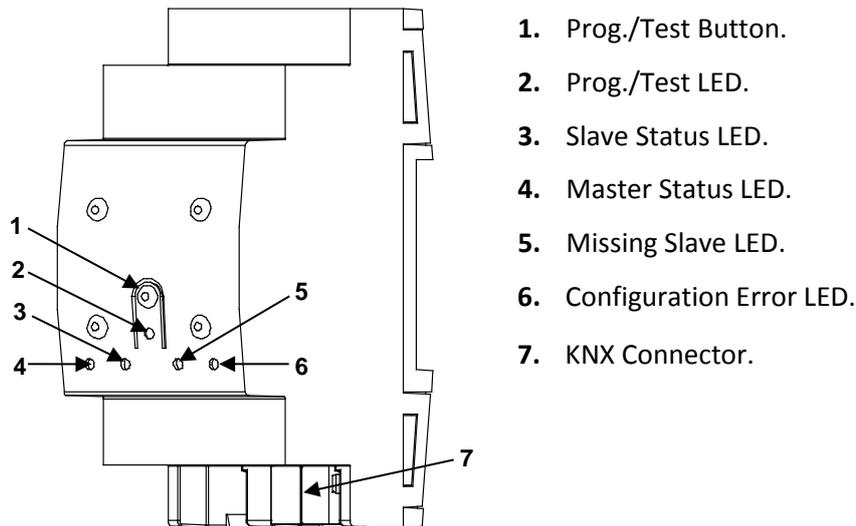


Figure 1. MCC - Element Diagram.

The main elements of the device are described next.

- **Test/Prog. Pushbutton (1):** a short press on this button sets the device into the programming mode, making the associated LED (2) light in red.  
*Note: if this button is held while plugging the device into the KNX bus, the device will enter into **safe mode**. In such case, the LED will blink in red.*
- **Master Status LED (3):** LED indicator that lights up when the device is running as a master.
- **Slave Status LED (4):** LED indicator that lights up when the device is running as a slave.
- **Missing Slave LED (5):** LED indicator that lights up when no slave MCC is found in the installation. Note that this does not prevent the master MCC from running normally.
- **Configuration Error LED (6):** LED indicator that will blink in red if, although one master MCC and one slave MCC have been found in the installation, their configurations do not match.

To get detailed information about the technical features of the device, as well as on the installation and security procedures, please refer to the corresponding **Datasheet**, bundled with the original package of the device and also available at [www.zennio.com](http://www.zennio.com)

## 2 FUNCTIONALITY

---

As indicated in the introduction, the aim of this device is performing an **in-line climate control of multiple hotel rooms** (or any other guestrooms, such as residential rooms or hospital rooms), where it is necessary to ensure that the system keeps running despite of occasional device faults.

The following sections describe the different roles that the device can assume and all the checks and notifications performed during the process.

### 2.1 CONFIGURATION CHECK

---

MCC is designed to **operate in pairs**: two MCCs with the same configuration are recommended to simultaneously work together in the same installation. Device redundancy is justified by how critical a fault in one of them would be, as the climate system of several rooms depends on it. However, **it is not mandatory to install two MCCs to make the system work**.

To have two devices working together properly it is necessary to configure them identically: same parameters, group addresses and associations. It is recommended to first configure a device and, after validating the configuration, **making an exact copy in the ETS project**, keeping the group addresses.

Still, the device will check that its configuration matches that of its analogue. Therefore, after each reboot both will exchange a **checksum value**, which depends on:

- Number and application program version.
- Parameters.
- Table of communication objects (flags and size of the communication objects).
- Group address table.
- Associations table.

Every time the master and the slave send the bus a **master / slave role confirmation notification** (see sections 2.2.1 and 2.2.2), they both will also re-send their **checksum values** through a communication object with the writing and transmission flags

enabled, in order to transmit and receive information through it. If **the received checksum is different from the one sent** (i.e., the one calculated internally):

- **Everything related to the thermostatic control will stop working:** no more sendings will take place, even if the device is working as a master (see section 2.2.1).
- **The role notification sending will stop** (master / slave).
- **The “Configuration error” communication object** will be transmitted with value ‘1’ every 30 seconds.
- **The checksum calculated internally** will be transmitted with a one-second delay, to make it possible to check at any time whether the two configurations still do not match.
- **All LEDs will turn off except the configuration error LED**, which will blink every one second.

The device will remain in the above state **until the expected checksum value** is received (i.e., a value that matches with the one internally calculated), after which the **device will reboot as if it had just been programmed**.

Once the device reboots, it will operate as if it had never detected the wrong configuration, thus, maintaining the same state it had before the configuration error detection. Then, it will apply everything specified in section 2.5.

After a restart, the object **“Configuration error”** is always transmitted with value ‘0’.

## 2.2 OPERATION ROLES

---

MCC may be operating as a master, as a slave or under an “undefined” state:

- The **master** aims at actually controlling the room climate.
- The **slave** remains waiting for a failure in the master, to take over the climate control.
- A device is operating as “**undefined**” when it has not been determined yet whether it must operate as a master or as a slave.

The device role will be notified via communication object (this process is detailed in following sections) and by two LEDs specifically included for this purpose: the LED corresponding to the current role lights up (**master LED indicator or slave LED indicator**); if there is no role assigned yet (see section 2.3), both LEDs remain off.

The integrated thermostats do not start operating until the device has a role assigned. When the device adopts a role, it will start performing the thermostat actions as if a restart had just taken place. Moreover, on the first execution, the **thermostat actions that may apply after an ETS download will also take place**.

The following sections describe the master and slave operating modes.

### 2.2.1 MASTER

---

The master handles the **climate control** by sending the control orders and states (communication objects) when required.

While the device is operating as a master, the master LED indicator is on.

#### 2.2.1.1 COMMUNICATION OBJECT NOTIFICATIONS

---

Periodically (time set by parameter), the master device sends a ‘1’ through the master notification communication object, and afterwards its internal checksum value.

If a **master device receives any value through the master notification object**, automatically switches to slave mode (and sends the corresponding slave notification).

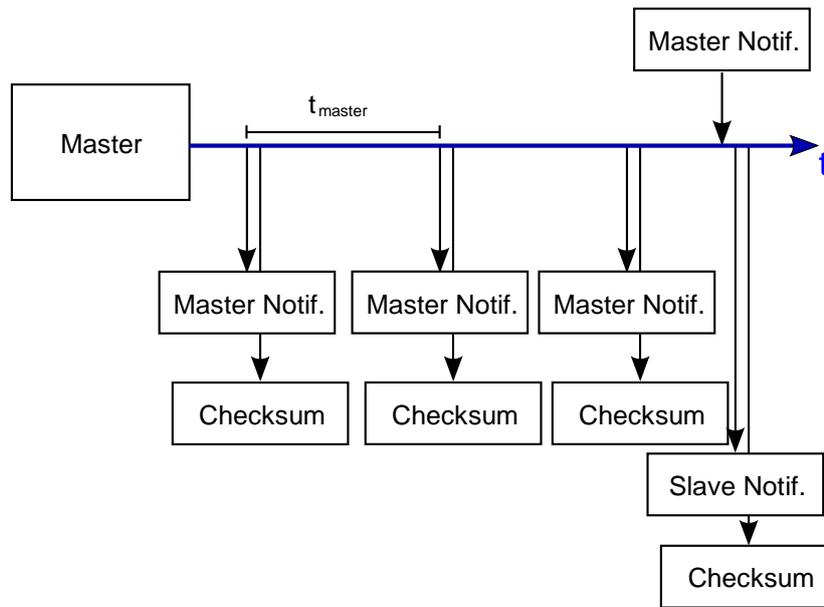


Figure 2 Master notification.

In order to avoid message collision, if a master notification is received (from another device) no later than 100 ms after sending a master notification, the device switches to the “**undefined**” state and it will start the role selection process (see section 2.3).

### 2.2.1.2 SLAVE NOTIFICATION RECEPTION

The master expects to receive periodically (through the slave notification object) a confirmation that the slave device is operating properly. If it doesn't receive **any value** through this object during **two times and a half** the slave notification time (set by parameter), the 'missing slave' LED indicator will turn on. And once **any value** is received through the communication object, the LED will turn off.

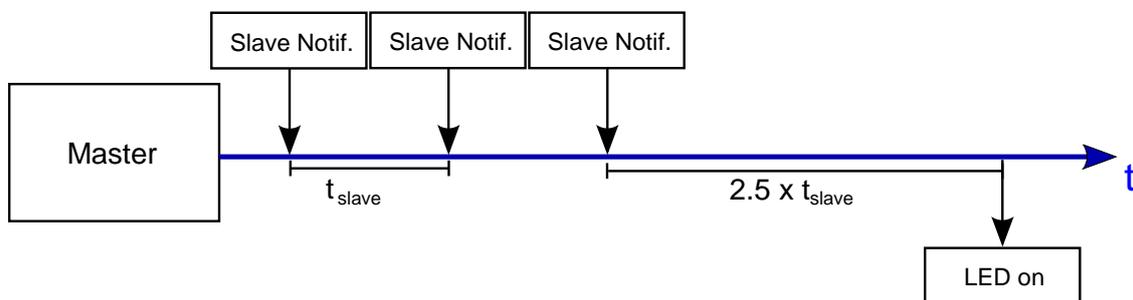


Figure 3 No-slave detection by the master.

**Note:** *the time to wait before reporting the ‘missing slave’ notification starts counting on the last reception of a slave notification (the first time, the count begins once the device adopts the master role).*

## 2.2.2 SLAVE

---

When MCC operates as a slave, its internal thermostats listen to the communication objects being written to the KNX bus, and update their states and communication objects. The main difference is that a **slave MCC doesn't send any communication objects (related to thermostats)** to the bus. Thus, master thermostats and slave thermostats will always have the same state (at least after a certain time, in case both MCCs were not put in operation simultaneously).

While the device is operating as a slave, the slave LED indicator is on.

### 2.2.2.1 SENDING READ REQUESTS

---

After a reboot, as long as MCC is operating as a slave (i.e., when the device state is not “master” nor “undefined”), the device may send **requests to read certain objects in order to update the statuses of its thermostats**, provided that such requests have been enabled by parameter. Specifically, the following communication objects are read:

- Transition Time: comfort to default mode
- Transition Time: standby to economy
- Comfort Setpoint Reset Time
- User Setpoint Control (Cooling and Heating)
- Comfort Setpoint (Cooling)
- Standby Setpoint (Cooling)
- Economy Setpoint (Cooling)
- Protection Setpoint (Cooling)
- Comfort Setpoint (Heating)
- Standby Setpoint (Heating)
- Economy Setpoint (Heating)
- Protection Setpoint (Heating)

- Comfort Lower Limit
- Comfort Upper Limit
- Hidden Offset Value
- Eco Mode: Lower Limit (Cooling)
- Eco Mode: Upper Limit (Heating)

These requests only apply to the enabled thermostats and to the linked objects.

**Notes:**

- *These read requests are **sent gradually**, to prevent the saturation of the KNX bus.*
- *To obtain the expected behaviour, the communication objects that are read-requested **must have the Writing, Transmission and Update flags** enabled.*
- *If a **role change** occurs while performing the read requests, the remaining responses will not be affected, as they will be sent as if the role change had not taken place.*

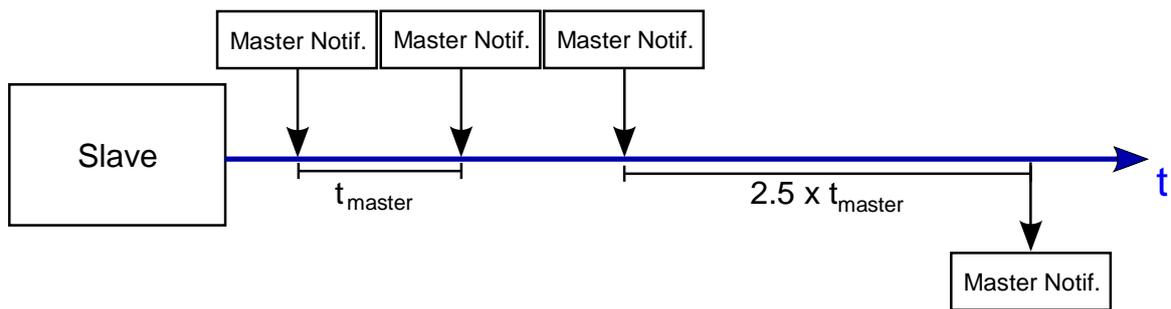
#### 2.2.2.2 OBJECT COMMUNICATION NOTIFICATIONS

---

Periodically (time set by parameter), the slave device sends a '1' through the slave notification communication object.

The reception of any value through the slave notification object will be ignored.

## 2.2.2.3 MASTER FAILURE DETECTION



**Figure 4** Master error detection by the slave.

In the slave role, the device permanently listens for the master notifications. After **two and a half times** the transmission time without receiving any master notification, **the slave device switches to the master role and notifies it by sending a “1” through the master notification communication object** (the role change notification object is also sent, with value ‘1’).

Thereafter the device will send the master notification periodically according to the configured cycle time

## 2.3 MASTER/SLAVE ROLE SELECTION

The device role is automatically assigned. When a device is under the “undefined” state, it starts the role selection algorithm:

1. A **random wait time** is calculated, greater enough than the master notification sending time.
2. The device remains waiting during the above time.
  - a. If nothing is received, it adopts the **master role** and starts sending the master notification object.
  - b. If before the end of the time count any value is received through the master notification object, the device adopts the **slave role** and starts sending the slave notification object.

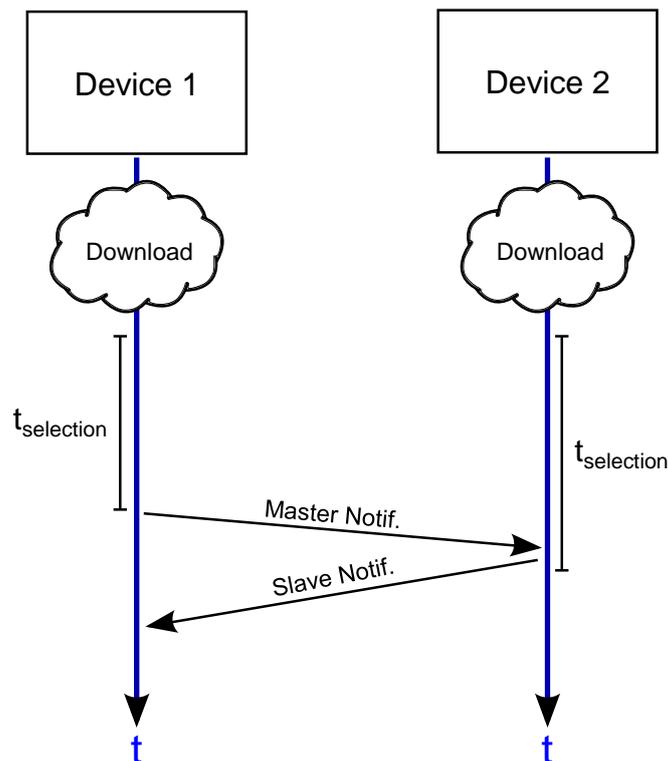


Figure 5 Role selection.

## 2.4 LED NOTIFICATIONS

---

The following table sums up the different LED notifications, which have been explained in the previous sections:

LED	Function	Description
<b>Master</b>	Notifying that the device is operating as a master.	It remains on since the device adopts the master role, and off in any other situation.
<b>Slave</b>	Notifying that the device is operating as a slave.	It remains on since the device adopts the slave role, and off in any other situation.
<b>Configuration Error</b>	Notifying that the pair MCC has a different configuration.	It starts blinking every one second if the checksum value received from the bus does not match the expected value.
<b>Slave Missing</b>	Notifying the absence of a slave MCC.	Will turn on after a certain time without receiving slave notifications, to indicate the absence of the slave MCC.

Table 1 LED notifications.

## 2.5 INITIAL STATES

When MCC has just been programmed, it always starts under the “undefined state”. At that point, the selection algorithm (see section 2.3) begins, the result of which will determine whether the device switches to master or to slave.

**Note:** as stated in section 2.3, MCC may return to the “undefined” if a master notification is received from the bus immediately after having sent an analogous notification. To prevent collisions between two MCCs opting to behave as masters, the “undefined” state will be adopted and the role selection algorithm will start.

Should a bus power failure take place before the selection algorithm has concluded, MCC will retry the role selection once it recovers from the power failure.

On the other hand, if a bus failure takes place once MCC had a role already assigned:

- If it was working as a slave, it will reboot as such and operate normally.
- If it was working as a master, it will reboot as such but will first ensure that no other master notifications are received during 1.5 times the master notification period.

The following table and graphs sum up all the above:

Initial Mode	Event	Next Mode	Actions to perform
Undefined	Download	Undefined	Beginning of the selection algorithm.
	Reboot		
Slave	Download	Slave	Slave notification and normal operation.
	Reboot		
Master	Download	Undefined	Beginning of the selection algorithm.
	Reboot	Master	Wait for 1.5x the master notification time. <ul style="list-style-type: none"> <li>- Master notifications received → switch to slave (and notify it<sup>1</sup>).</li> <li>- No master notification received → remain as master (and notify it).</li> </ul>

**Table 2** Initial states after ETS download or reboot (power failure).

<sup>1</sup> READ requests will also be transmitted, if configured.



Figure 6 Device state after ETS download.

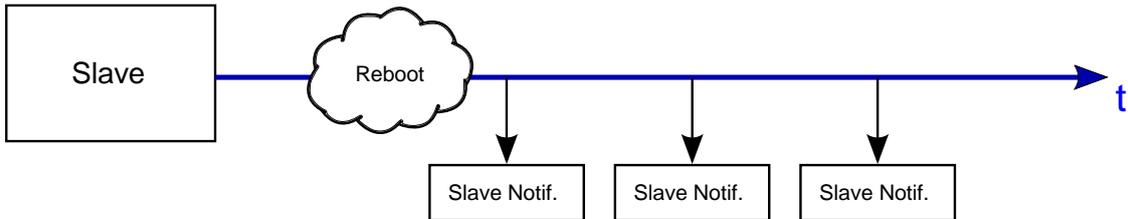


Figure 7 State after reboot (as slave).

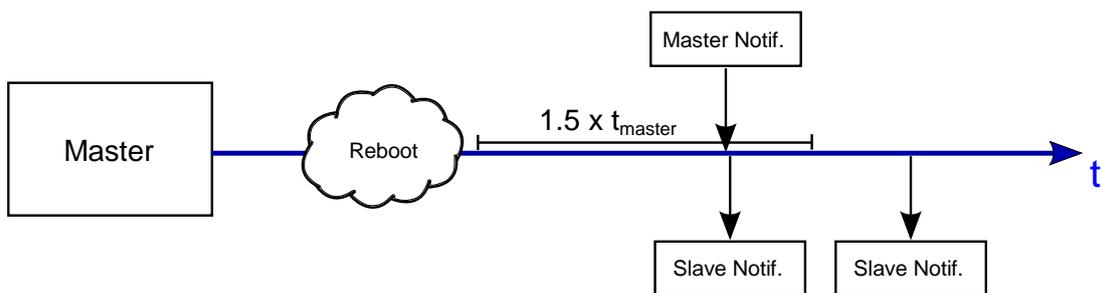
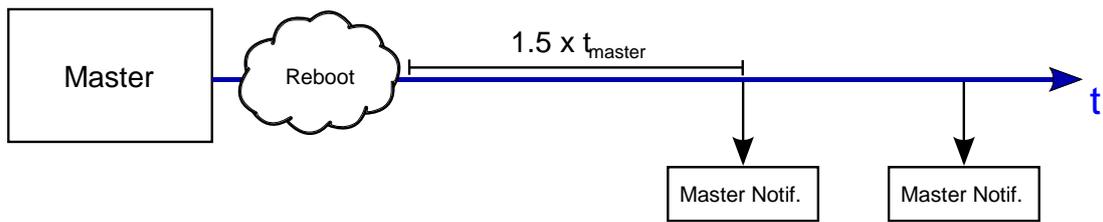


Figure 8 State after reboot (as master).

## 3 CONFIGURATION

### 3.1 GENERAL

#### ETS PARAMETERISATION

After importing the corresponding database in ETS and adding the device into the topology of the desired project, the configuration process begins by right-clicking into the device and selecting *Edit parameters*.

The “General” parameter screen is the only shown by default.

System Configuration: Master-Slave

Master Notification Period: 4 x 1 s.

Slave Notification Period: 10 x 1 s.

Send read requests after a reboot (only for slave mode):

---

Hospitality Thermostat 1

Hospitality Thermostat 2

Hospitality Thermostat 3

Hospitality Thermostat 4

Hospitality Thermostat 5

Hospitality Thermostat 6

Hospitality Thermostat 7

Hospitality Thermostat 8

Hospitality Thermostat 9

Hospitality Thermostat 10

Hospitality Thermostat 11

Hospitality Thermostat 12

Hospitality Thermostat 13

Hospitality Thermostat 14

Figure 9. General.

It contains the following parameters:

- **System Configuration:** sets whether the system comprises two MCCs, one operating as a master and the other as a slave ("Master-Slave", default option) or if it consists of just one device ("Only one device").

When selecting the "Master-Slave" configuration, the following parameters show up:

- **Master Notification Time:** sets the sending period of the master notifications, through which MCC informs (when applicable) that it is working in the master mode. Range: 1-255 s (4 s by default).
- **Slave Notification Time:** sets the sending period of the slave notifications, through which MCC informs (when applicable) that it is working in the slave mode. Range: 1-255 s (4 s by default).
- **Send read requests after a reboot (only for slave mode):** enables the sending of read requests of certain objects by the slave after a reboot so it can update its state to match that of the master. It is enabled by default.

In this case, the default communication objects are:

- **"Master":** one-bit object which is sent periodically (according to parameter "**Master Notification Time**") to indicate the device is operating in master mode.
- **"Slave":** one-bit object which is sent periodically (according to parameter "**Slave Notification Time**") to indicate the device is operating in slave mode.
- **"Role Switch":** one-bit object which is sent (with the value '1') when a role change occurs.
- **"Checksum":** two-byte object through which the internal checksum value is sent. The analogous value from the additional MCC is also expected to be received through this object (to ensure their configurations match).
- **"Configuration Error":** one-bit object which is sent (periodically) with the value '1' when the master and the slave do not share the same configuration (checksum mismatch). Once solved, it will be sent with the value '0'.

When selecting "Only one device" system configuration, the following parameters show up (Figure 10):

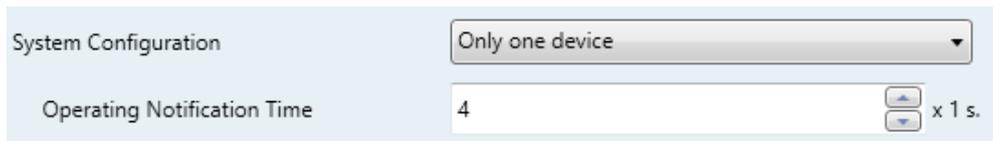


Figure 10 General – System Configuration: Only one device.

- **Operating Notification Time:** sets the sending period of the operating notifications, which indicate that the device is working. Range: 1-255 s (4 s by default).

In this case, the default communication objects are:

- **“Operating Notification”:** one-bit object through which the above notifications are periodically sent.

On the other hand, as shown in Figure 9, **fourteen checkboxes** (one per each Hospitality thermostat) are provided to let the integrator select how many of the fourteen thermostats are required. Additional entries to configure them will be included in the tab menu on the left depending on the number of active thermostats.

## 3.2 HOSPITALITY THERMOSTAT

---

As described in previous sections, MCC includes **fourteen instances of the Hospitality Thermostat**, which can be enabled and customised independently.

Please refer to the specific manual **“Hospitality Thermostat in MCC”** (available at the Zennio homepage, [www.zennio.com](http://www.zennio.com)) for detailed information about the functionality and the configuration of the involved parameters.

## ANNEX I. COMMUNICATION OBJECTS

- “Functional range” shows the values that, with independence of any other values permitted by the bus according to the object size, may be of any use or have a particular meaning because of the specifications or restrictions from both the KNX standard or the application program itself.

Number	Size	I/O	Flags	Data type (DPT)	Functional Range	Name	Function
1	1 Bit	I	C T - W -	DPT_Trigger	0/1	Master	Notification of master mode operation
	1 Bit		C T - - -	DPT_Trigger	0/1	Operating Notification	Notification that the device is working properly
2	1 Bit	I	C T - W -	DPT_Trigger	0/1	Slave	Notification of slave mode operation
3	1 Bit		C T - - -	DPT_Trigger	0/1	Role Switch	Role switch notification
4	2 Bytes	I	C T - W -	DPT_Value_2_Ucount	0 - 65535	Checksum	Configuration checksum
5	1 Bit	O	C T R - -	DPT_Alarm	0/1	Configuration Error	0 = No Error; 1 = Configuration Error
[6 + 54(x-1)]	1 Byte	I	C - - W -	DPT_SceneControl	0-63; 128-191	[HTx] [A] Scene Input	Scene Value
[7 + 54(x-1)]	2 Bytes	I	C - - W -	DPT_Value_Temp	-273.00 - 670760.00	[HTx] [A] Temperature Source 1	External Sensor Temperature
[8 + 54(x-1)]	2 Bytes	I	C - - W -	DPT_Value_Temp	-273.00 - 670760.00	[HTx] [A] Temperature Source 2	External Sensor Temperature
[9 + 54(x-1)]	2 Bytes	O	C T R - -	DPT_Value_Temp	-273.00 - 670760.00	[HTx] [A] Room Temperature	Current temperature
[10 + 54(x-1)]	1 Byte	I	C - - W -	DPT_HVACMode	1=Comfort 2=Standby 3=Economy 4=Building Protection	[HTx] [D] Special Mode	1-byte HVAC Mode
[11 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Trigger	0/1	[HTx] [D] Special Mode: comfort	0 = Nothing; 1 = Trigger
	1 Bit	I	C - - W -	DPT_Switch	0/1	[HTx] [D] Special Mode: comfort	0 = Off; 1 = On
[12 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Trigger	0/1	[HTx] [D] Special Mode: standby	0 = Nothing; 1 = Trigger
	1 Bit	I	C - - W -	DPT_Switch	0/1	[HTx] [D] Special Mode: standby	0 = Off; 1 = On
[13 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Trigger	0/1	[HTx] [D] Special Mode: economy	0 = Nothing; 1 = Trigger
	1 Bit	I	C - - W -	DPT_Switch	0/1	[HTx] [D] Special Mode: economy	0 = Off; 1 = On
[14 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Trigger	0/1	[HTx] [D] Special Mode: protection	0 = Nothing; 1 = Trigger
	1 Bit	I	C - - W -	DPT_Switch	0/1	[HTx] [D] Special Mode: protection	0 = Off; 1 = On
[15 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Occupancy	0/1	[HTx] [C] Presence Detector (input)	0 = Not Occupied; 1 = Occupied
[16 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Bool	0/1	[HTx] [C] Sold/Unsold Room (input)	0 = Unsold; 1 = Sold
[17 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 1 (input)	0 = Closed; 1 = Open
	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 1 (input)	0 = Open; 1 = Closed
[18 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 2 (input)	0 = Closed; 1 = Open

	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 2 (input)	0 = Open; 1 = Closed
[19 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 3 (input)	0 = Closed; 1 = Open
	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 3 (input)	0 = Open; 1 = Closed
[20 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 4 (input)	0 = Closed; 1 = Open
	1 Bit	I	C - - W -	DPT_Window_Door	0/1	[HTx] [D] Window Status 4 (input)	0 = Open; 1 = Closed
[21 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Enable	0/1	[HTx] [D] Enable Window Status	0 = Disabled; 1 = Enabled
[22 + 54(x-1)]	1 Byte	O	C T R - -	DPT_HVACMode	1=Comfort 2=Standby 3=Economy 4=Building Protection	[HTx] [D] Special Mode Status	1-byte HVAC Mode
[23 + 54(x-1)]	1 Bit	O	C T R - -	DPT_Switch	0/1	[HTx] [D] Comfort Mode Status	0 = Off; 1 = On
[24 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Enable	0/1	[HTx] [D] Thermostat Lock	0 = Locked; 1 = Unlocked
	1 Bit	I	C - - W -	DPT_Enable	0/1	[HTx] [D] Thermostat Lock	0 = Unlocked; 1 = Locked
[25 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_TimePeriodSec	0 - 65535	[HTx] [C] Transition Time: comfort to default mode	Seconds (0 = Disabled)
	2 Bytes	I	C T - W U	DPT_TimePeriodMin	0 - 65535	[HTx] [C] Transition Time: comfort to default mode	Minutes (0 = Disabled)
	2 Bytes	I	C T - W U	DPT_TimePeriodHrs	0 - 65535	[HTx] [C] Transition Time: comfort to default mode	Hours (0 = Disabled)
[26 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_TimePeriodSec	0 - 65535	[HTx] [C] Transition Time: standby to economy	Seconds (0 = Disabled)
	2 Bytes	I	C T - W U	DPT_TimePeriodMin	0 - 65535	[HTx] [C] Transition Time: standby to economy	Minutes (0 = Disabled)
	2 Bytes	I	C T - W U	DPT_TimePeriodHrs	0 - 65535	[HTx] [C] Transition Time: standby to economy	Hours (0 = Disabled)
[27 + 54(x-1)]	1 Bit	I	C - - W -	DPT_Reset	0/1	[HTx] [B] User Comfort Setpoint Reset	0 = Nothing; 1 = Reset
[28 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_TimePeriodSec	0 - 65535	[HTx] [C] Comfort Setpoint Reset Time	Seconds (0 = Disabled)
	2 Bytes	I	C T - W U	DPT_TimePeriodMin	0 - 65535	[HTx] [C] Comfort Setpoint Reset Time	Minutes (0 = Disabled)
	2 Bytes	I	C T - W U	DPT_TimePeriodHrs	0 - 65535	[HTx] [C] Comfort Setpoint Reset Time	Hours (0 = Disabled)
[29 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] User Setpoint Control (Cooling and Heating)	[-20°C, 100°C]
[30 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Comfort Setpoint (Cooling)	[-20°C, 100°C]
[31 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Standby Setpoint (Cooling)	[-20°C, 100°C]
[32 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Economy Setpoint (Cooling)	[-20°C, 100°C]
[33 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Protection Setpoint (Cooling)	[-20°C, 100°C]
[34 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Comfort Setpoint (Heating)	[-20°C, 100°C]
[35 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Standby Setpoint (Heating)	[-20°C, 100°C]
[36 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Economy Setpoint (Heating)	[-20°C, 100°C]
[37 + 54(x-1)]	2 Bytes	I	C T - W U	DPT_Value_Temp	-20 - 100	[HTx] [B] Protection Setpoint (Heating)	[-20°C, 100°C]
[38 + 54(x-1)]	2 Bytes	O	C T R - -	DPT_Value_Temp	-20 - 100	[HTx] [B] Real Setpoint Status	[-20°C, 100°C]
[39 + 54(x-1)]	2 Bytes	O	C T R - -	DPT_Value_Temp	-20 - 100	[HTx] [B] User Setpoint Status	[-20°C, 100°C]

[40 + 54(x-1)]	2 Bytes	I	<b>CT - W U</b>	DPT_Value_Temp	-20 - 100	[HTx] [D] Comfort Lower Limit	[-20°C, 100°C]
[41 + 54(x-1)]	2 Bytes	I	<b>CT - W U</b>	DPT_Value_Temp	-20 - 100	[HTx] [D] Comfort Upper Limit	[-20°C, 100°C]
[42 + 54(x-1)]	1 Bit	I	<b>C - - W -</b>	DPT_Switch	0/1	[HTx] [D] Hidden Offset On/Off	0 = Off; 1 = On
[43 + 54(x-1)]	2 Bytes	I	<b>CT - W U</b>	DPT_Value_Tempd	-20 - 100	[HTx] [D] Hidden Offset Value	[-20°C, 100°C]
[44 + 54(x-1)]	1 Bit	I	<b>C - - W -</b>	DPT_Heat_Cool	0/1	[HTx] [A] Mode	0 = Cooling; 1 = Heating
[45 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Heat_Cool	0/1	[HTx] [A] Mode Status	0 = Cooling; 1 = Heating
[46 + 54(x-1)]	1 Bit	I	<b>C - - W -</b>	DPT_Switch	0/1	[HTx] [A] On/Off	0 = Off; 1 = On
[47 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [A] On/Off Status	0 = Off; 1 = On
[48 + 54(x-1)]	1 Byte	O	<b>CTR - -</b>	DPT_Scaling	0% - 100%	[HTx] [Cooling] Control Variable	PI Control (Continuous)
[49 + 54(x-1)]	1 Byte	O	<b>CTR - -</b>	DPT_Scaling	0% - 100%	[HTx] [Heating] Control Variable	PI Control (Continuous)
[50 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Cooling] Control Variable	2-Point Control
	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Cooling] Control Variable	PI Control (PWM)
[51 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Heating] Control Variable	2-Point Control
	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Heating] Control Variable	PI Control (PWM)
[52 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Cooling] Additional Cool	Temp >= (Setpoint+Band) => "1"
[53 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Heating] Additional Heat	Temp <= (Setpoint-Band) => "1"
[54 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Cooling] PI State	0 = PI signal 0%; 1 = PI signal greater than 0%
[55 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Switch	0/1	[HTx] [Heating] PI State	0 = PI signal 0%; 1 = PI signal greater than 0%
[56 + 54(x-1)]	1 Bit	O	<b>CTR - -</b>	DPT_Boo1	0/1	[HTx] [D] Eco Mode Notification	0 = Out of eco range; 1 = Setpoint in eco range
[57 + 54(x-1)]	1 Byte	O	<b>CTR - -</b>	DPT_Scaling	0% - 100%	[HTx] [D] Eco Mode Ratio	Percentage of time working in eco range
[58 + 54(x-1)]	2 Bytes	I	<b>CT - W U</b>	DPT_Value_Temp	-20 - 100	[HTx] [D] Eco Mode: Lower Limit (Cooling)	Lower value for the ecological range
[59 + 54(x-1)]	2 Bytes	I	<b>CT - W U</b>	DPT_Value_Temp	-20 - 100	[HTx] [D] Eco Mode: Upper Limit (Heating)	Upper value for the ecological range

Join and send us your inquiries  
about Zennio devices:  
<http://zennioenglish.zendesk.com>

**Zennio Avance y Tecnología S.L.**  
C/ Río Jarama, 132. Nave P-8.11  
45007 Toledo (Spain).

*Tel. +34 925 232 002.*

*Fax. +34 925 337 310.*

*www.zennio.com*

*info@zennio.com*



RoHS