Electronic Intelligent Controls, S.L. | 08830 Sant Boi de Llobregat | Tel.: +34 93 652 55 21 | www.e-controls.es
C/Múrcia, 35 F | Barcelona, Spain | Fax: +34 93 652 55 22 | info@e-controls.es

# e-Multisensor® LON TP/FT-10

*DETECTOR DE MOVIMIENTO, LUMINOSIDAD Y TEMPERATURA PARA REDES LONWORKS®*
*Ref: MS.623000-000*

# Perfil Funcional

*Versión 0.1.0*

Este documento describe las variables de red y los parámetros de configuración del producto que forman su interface de red LonWorks®. La aplicación está formada por objetos lógicos (perfiles funcionales) de acuerdo con las Directrices de Interoperabilidad de LONMARK™.

Resource Files versión 1.1

## Perfiles Funcionales del producto

| Cantidad | Código | Perfil Funcional | Versión |
|---|---|---|---|
| 1 | 0000 | Node Object | 1.0 |
| 1 | 1010 | Light Sensor | 1.0 |
| 1 | 1060 | Occupancy Sensor | 1.0 |
| 1 | 1040 | Temperature Sensor | 1.0 |
| 1 | 3050 | Constant Light Controller | 1.0 |
| 1 | 3071 | Occupancy Controller | 1.0 |

e-controls
electronic intelligent controls, s.l.

## Perfil Funcional:

# Node Object

## 08504 v1.0

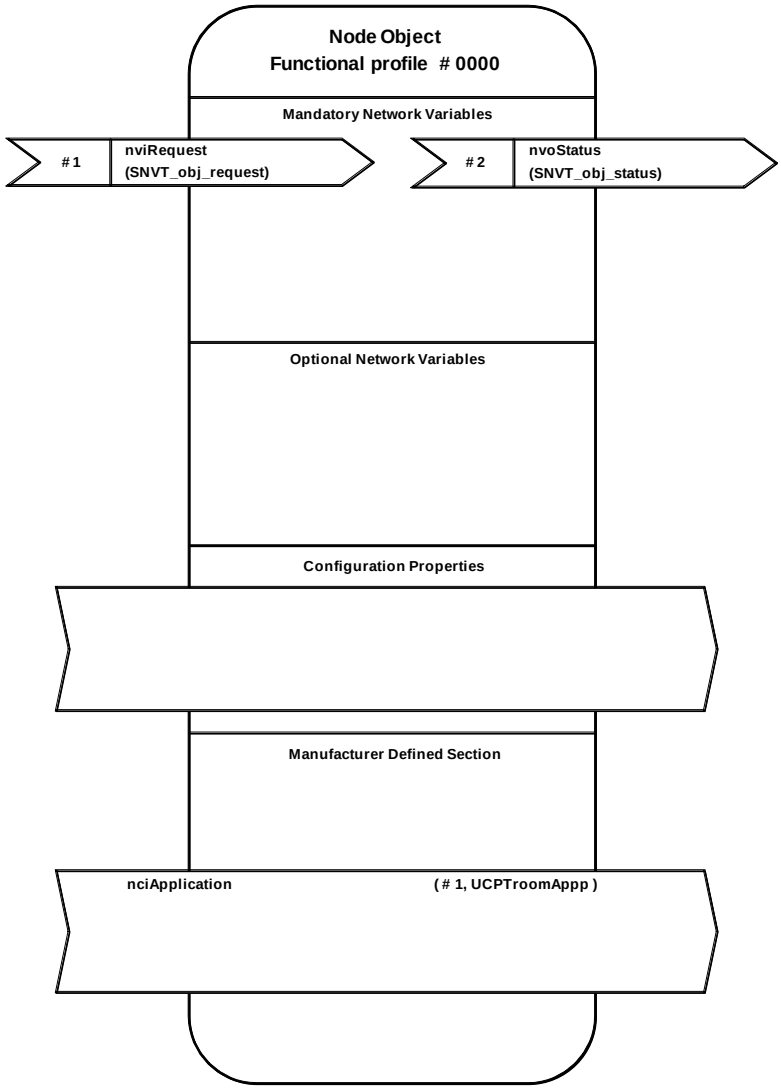
# *Contenido*

**1. Descripción ..... 3**
**2. Interfaz de red ..... 3**
**3. Variables de red ..... 4**
3.1. nviRequest ..... 4
3.2. nvoStatus ..... 5
**4. Histórico de revisiones ..... 6**

# 1. Descripción

El perfil *Node Object* es un bloque funcional especial que usan las herramientas de red para testear y gestionar todos los perfiles funcionales del equipo.

# 2. Interfaz de red



# 3. Variables de red

## 3.1. nviRequest

```
network input SNVT_obj_request nviRequest;
```

Esta variable proporciona un mecanismo para pedir una operación o un modo para uno de los perfiles funcionales del equipo.

### *Tipo*

***SNVT_obj_request***

```
typedef struct
{
      unsigned long object_id;
      object_request_t object_request;
} SNVT_obj_request;
```

### *Margen de valores*

`object_id`

| | |
|---|---|
| 0…12 | índice al perfil requerido |
| resto de valores | inválido |

`object_request`

```
typedef enum object_request_t {
      /*  0 */    RQ_NORMAL,
      …
      /*  2 */    RQ_UPDATE_STATUS,
      …
      /*  5 */    RQ_REPORT_MASK,
      …
      /* -1 */    RQ_NUL = -1
} object_request_t;
```

| | |
|---|---|
| `RQ_NORMAL` | Si el perfil especificado estuviera deshabilitado, la petición cancela este estado y devuelve el perfil al estado normal. |
| `RQ_UPDATE_STATUS` | Solicita que el estado del perfil especificado se envíe por la variable `nvoStatus`. |
| `RQ_REPORT_MASK` | Solicita que se envíe por la variable `nvoStatus` una *máscara* de status con los bits de status implementados en el perfil indicado. |

### *Valor por defecto*

| | |
|---|---|
| { 0, 0 } | Requerir el estado normal al perfil `NodeObject`. |

## 3.2. nvoStatus

```
network output SNVT_obj_status nvoStatus;
```

Esta variable muestra el estado de cualquier perfil funcional del equipo.

### *Tipo*

***SNVT_obj_status***

```
typedef struct
{
      unsigned long object_id;
      unsigned invalid_id           :1;         // bit0
      unsigned invalid_request      :1;         // bit1
      ...
      unsigned report_mask          :1;         // bit11
      ...
} SNVT_obj_status;
```

### *Margen de valores*

Cualquier valor dentro de los límites de ***SNVT_obj_status***.

### *Valor por defecto*

{ 0, [0…0] }

### *Cuando se transmite*

- Después de reset.
- Cuando se recibe una solicitud en la variable `nviRequest`.

### *Tipo de servicio por defecto*

No especificado, aunque se recomienda *acknowledged*. También se recomienda obtener su valor mediante encuesta (*polling*).

e-controls
electronic intelligent controls, s.l.

## Perfil Funcional:

# Light Sensor

## 10501 r1.0

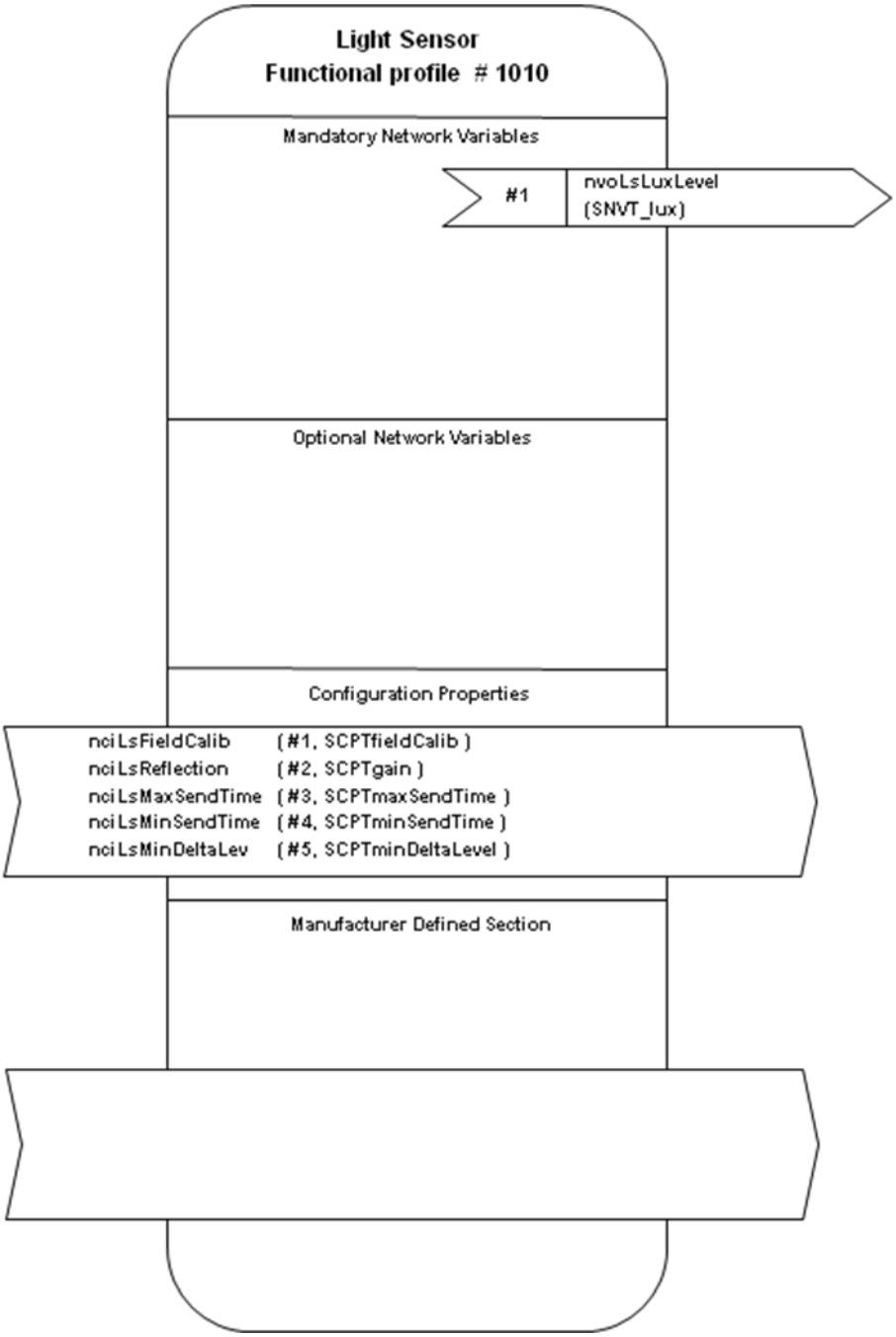
# Contenido

**1. Descripción ..... 3**
**2. Interfaz de red ..... 3**
**3. Variables de red ..... 4**
3.1. nvoLsLuxLevel ..... 4
**4. Variables de configuración ..... 5**
4.1. nciLsFieldCalibr ..... 5
4.2. nciLsReflection ..... 6
4.3. nciLsMaxSendTime ..... 7
4.4. nciLsMinSendTime ..... 7
4.5. nciLsMinDeltaLev ..... 8

# 1. Descripción

El objeto *Light Sensor* se utiliza en dispositivos con sensor de luminosidad cuya salida indica el nivel de luz ambiente de la estancia.

# 2. Interfaz de red



# 3. Variables de red

## 3.1. nvoLsLuxLevel

```
network output SNVT_lux nvoLsLuxLevel;
```

Esta variable proporciona el nivel de luz ambiente de la zona donde está instalado el sensor de luz.

### Tipo

***SNVT_lux***

```
typedef unsigned long SNVT_lux;
```

### Margen de valores

0 .. 1000 (0 .. 1000 lux, con resolución 1 lux)

### Valor por defecto

{ 0 } 0 lux

# 4. Variables de configuración

## *4.1. nciLsFieldCalibr*

```
network input cp SCPTfieldCalib nciLsFieldCalibr;
```

Esta propiedad de configuración se utiliza para ajustar el factor de ganancia del sensor de luz al entorno. Para ajustar el factor de ganancia se debe asignar a esta variable de configuración el nivel de lux leído con un luxómetro.

### *Tipo*

***SCPTfieldCalib, derivado de SNVT_lux.***

### *Margen de valores*

| | |
|---|---|
| 1 .. 1000 | (1 .. 1000 lux, con resolución 1 lux) |

### *Valor por defecto*

| | |
|---|---|
| { 0 } | Indica que el factor de ganancia del sensor de luz no está ajustado al entorno. Es el valor por defecto de fábrica. |

## 4.2. nciLsReflection

```
network input cp SCPTgain nciLsReflection;
```

Esta propiedad de configuración define el nivel de reflexión de la superficie situada debajo del sensor y se utiliza para ajustar el factor de ganancia del sensor de luz al entorno.

El factor de ganancia se ajusta a partir del porcentaje de reflexión de la superficie situada debajo del sensor. Por ejemplo si el porcentaje de reflexión es del 20% se debe asignar al campo `multiplier` el valor de 100 y al campo `divisor` el valor de 20

También se puede ajustar el factor de ganancia sin conocer el porcentaje de reflexión de la superficie si se dispone de un luxómetro. En este caso en el campo `divisor` se debe introducir el valor de lux leído por el sensor del equipo y proporcionado por `nvoLsLuxLevel` y en el campo `multiplier` el valor de lux leído con el luxómetro.

NOTA: Esta propiedad de configuración no tiene efecto sobre el equipo si la propiedad de configuración `nciLsFieldCalibr` tiene un valor diferente de 0, es decir si el factor de ganancia ya ha sido ajustado mediante `nciLsFieldCalibr`.

### *Tipo*

***SCPTgain, derivado de SNVT_muldiv.***

```
typedef struct
{
     unsigned long multiplier;
     unsigned long divisor;
} SNVT_switch;
```

### *Margen de valores*

`multiplier (lux medidos por el luxómetro)`

1 .. 1000 (1 .. 1000 lux, con resolución 1 lux)

`divisor (lux medidos por el sensor)`

1.. 1000 (1 .. 1000 lux, con resolución 1 lux)

*Valor por defecto*

{1, 1} Ganancia 1

## 4.3. nciLsMaxSendTime

```
network input cp SCPTmaxSendTime nciLsMaxSendTime;
```

Esta propiedad de configuración se utiliza para controlar el tiempo máximo que puede pasar antes que el objeto envíe automáticamente el valor actual de la variable de salida `nvoLsLuxLevel`. Esto proporciona un periodo de transmisión que asegura que el objeto está activo. Se puede deshabilitar su funcionamiento definiendo el valor a cero.

`Tipo`

***SCPTmaxSendTime, derivado de SNVT_time_sec.***

*Margen de valores:*

0 - 6553,4s (0 .. 6553,4 segundos, con resolución 1 décima de segundo)

*Valor por defecto:*

{600} El valor por defecto es 1 minuto

## 4.4. nciLsMinSendTime

```
network input cp SCPTminSendTime nciLsMinSendTime;
```

Esta propiedad de configuración se utiliza para definir el tiempo mínimo que debe pasar entre el envío de dos valores diferentes de la variable. Esta propiedad permite limitar la cantidad de valores a enviar por unidad mínima de tiempo.

`Tipo`

***SCPTminSendTime, derivado de SNVT_time_sec.***

*Margen de valores:*

0 - 6553,4s (0 .. 6553,4 segundos, con resolución 1 décima de segundo)

### *Valor por defecto*

{10} El valor por defecto es de 1 segundo

## *4.5. nciLsMinDeltaLev*

```
network input cp SCPTminDeltaLevel nciLsMinDeltaLev;
```

Esta propiedad de configuración se utiliza para definir el valor mínimo que debe variar la salida `nvoLsLuxLevel` para transmitir de nuevo su valor por la red.

### `Tipo`

***SCPTminDeltaLevel, derivado de SNVT_lev_cont.***

### *Margen de valores:*

El rango es de 0.0% - 100.0% en pasos de 0.5%

### *Valor por defecto*

{4} El valor por defecto es del 2%

e-controls
electronic intelligent controls, s.l.

## Perfil Funcional:

# Occupancy Sensor

## 10501 r1.0

# Contenido

**1. Descripción ..... 3**
1.1. Funcionamiento ..... 3
**2. Interfaz de red ..... 4**
**3. Variables de red ..... 5**
3.1. nvoOsOccup ..... 5
**4. Variables de configuración ..... 6**
4.1. nciOsDebounce ..... 6
4.2. nciOsHeartBeat ..... 6
4.3. nciOsPIRthreshold ..... 7
4.4. nciOsIndicator ..... 8

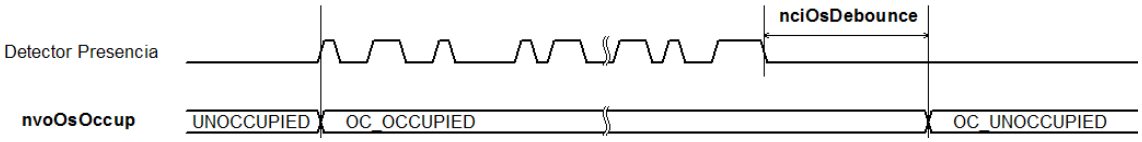
# 1. Descripción

El objeto *Occupancy Sensor* se utiliza en dispositivos con sensores de movimiento cuya salida indica el estado *Ocupado/No-Ocupado* de una estancia. El paso a estado *Ocupado* se produce por un flanco de activación del sensor, mientras que el paso al estado No-Ocupado se produce al finalizar el tiempo establecido en la variable de configuración `nciOsDebounce`.

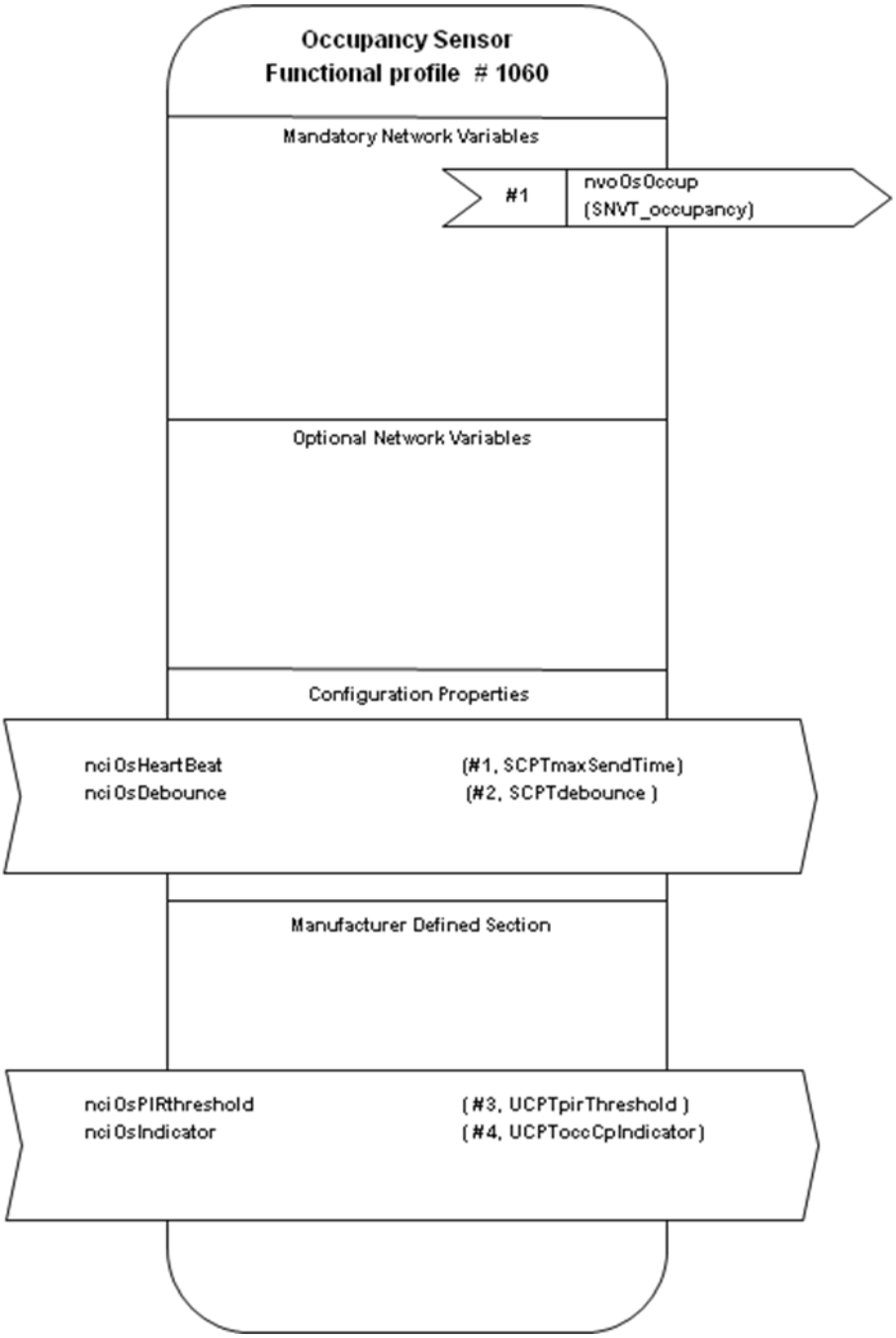
Su variable de salida se suele conectar a un controlador, que se encarga de realizar una acción en función de la Ocupación, como puede ser activar una luminaria o ajustar la consigna de un climatizador a un nivel predeterminado.

## *1.1. Funcionamiento*

Un flanco de subida del sensor provoca el paso a *Ocupado*, mientras que un flanco de bajada activa un temporizador interno que, al finalizar, provoca el paso a No-Ocupado. Dicho temporizador se recarga con cada flanco de subida.



# 2. Interfaz de red



# 3. Variables de red

## 3.1. nvoOsOccup

```
network output SNVT_occupancy nvoOsOccup;
```

Esta variable proporciona el estado de *Ocupación* de la zona donde está instalado el sensor de movimiento.

### Tipo

SNVT_occupancy

```
typedef occup_t SNVT_occupancy;
```

### Margen de valores

```
typedef enum occup_t
{
     /*  0 */  OC_OCCUPIED,
     /*  1 */  OC_UNOCCUPIED,
     …
     /* -1 */  OC_NUL = -1
} occup_t;
```

### Valor por defecto

```
{-1}            OC_NUL
```

# 4. Variables de configuración

## 4.1. nciOsDebounce

```
network input cp SCPTdebounce nciOsDebounce;
```

Esta propiedad de configuración establece el periodo de tiempo que debe pasar desde que el sensor deja de detectar ocupación hasta que la variable nvoOsOccup pasa al valor *No-Ocupado*.

### *Tipo*

***SCPTdebounce, derivado de SNVT_time_sec.***

### *Margen de valores*

| | |
|---|---|
| 0 - 6553,4s | (0 .. 6553,4 segundos, con resolución 1 décima de segundo) |

### *Valor por defecto*

| | |
|---|---|
| {3} | 300 milisegundos |

## 4.2. nciOsHeartBeat

```
network input cp SCPTmaxSendTime nciOsHeartBeat
```

Esta propiedad de configuración define el periodo de repetición de envío de valores de variables de red a través del bus de comunicaciones. Su propósito es el de asegurar que el sensor está activo y permitir a un controlador tener múltiples sensores conectados a la misma variable.

### *Tipo*

***SCPTmaxSendTime, derivado de SNVT_time_sec.***

### *Margen de valores*

0 – 6553,4 con incrementos de 0.1 s

### *Valor por defecto*

{1200} 2 minutos

## *4.3. nciOsPIRthreshold*

```
network input cp UCPTpirThreshold nciOsPIRthreshold
```

Esta propiedad se utiliza para configurar el nivel de sensibilidad del sensor de movimiento para detectar un estado de *Ocupación* de la zona.

### *Tipo*

***UCPTpirThreshold, derivado del enumerado sensitivity_level_t.***

### *Margen de valores*

```
typedef enum sensitivity_level_t {
     LEVEL_1_MIN  = 0,
     LEVEL_2      = 1,
     LEVEL_3      = 2,
     LEVEL_4      = 3,
     LEVEL_5      = 4,
     LEVEL_6      = 5,
     LEVEL_7      = 6,
     LEVEL_8      = 7,
     LEVEL_9      = 8,
     LEVEL_10     = 9,
     LEVEL_11     = 10,
     LEVEL_12     = 11,
     LEVEL_13     = 12,
     LEVEL_14     = 13,
     LEVEL_15     = 14,
     LEVEL_16_MAX = 15,
     LEVEL_NUL   = -1}
 sensitivity_level_t;
```

`LEVEL_1_MIN`: mínima sensibilidad
`LEVEL_16_MAX`: máxima sensibilidad
`LEVEL_NUL`: el nivel de sensibilidad se deshabilita y el sensor deja de indicar el valor de `OCCUPIED`.

### *Valor por defecto*

{LEVEL_12}

## 4.4. nciOsIndicator

```
network input cp UCPToccCpIndicator nciOsIndicator
```

Esta propiedad de configuración indica si el indicador de ocupación al detector movimiento está active o no. El indicador de ocupación suele ser un dispositivo visual como un Led.

Esta propiedad de configuración define si el indicador Led del detector de movimiento se debe iluminar al detectar un movimiento considerado como válido.

### *Tipo*

***UCPToccCpIndicator, derivado de UNVT_occCpIndicator (boolean_t).***

### *Margen de valores*

0: El indicador Led está deshabilitado
1: El indicador Led está habilitado (se activa al detectar movimiento)

### *Valor por defecto*

{1} Indicador Led habilitado

e-controls
electronic intelligent controls, s.l.

## Perfil Funcional:

# Temperature Sensor

## 10501 r1.0

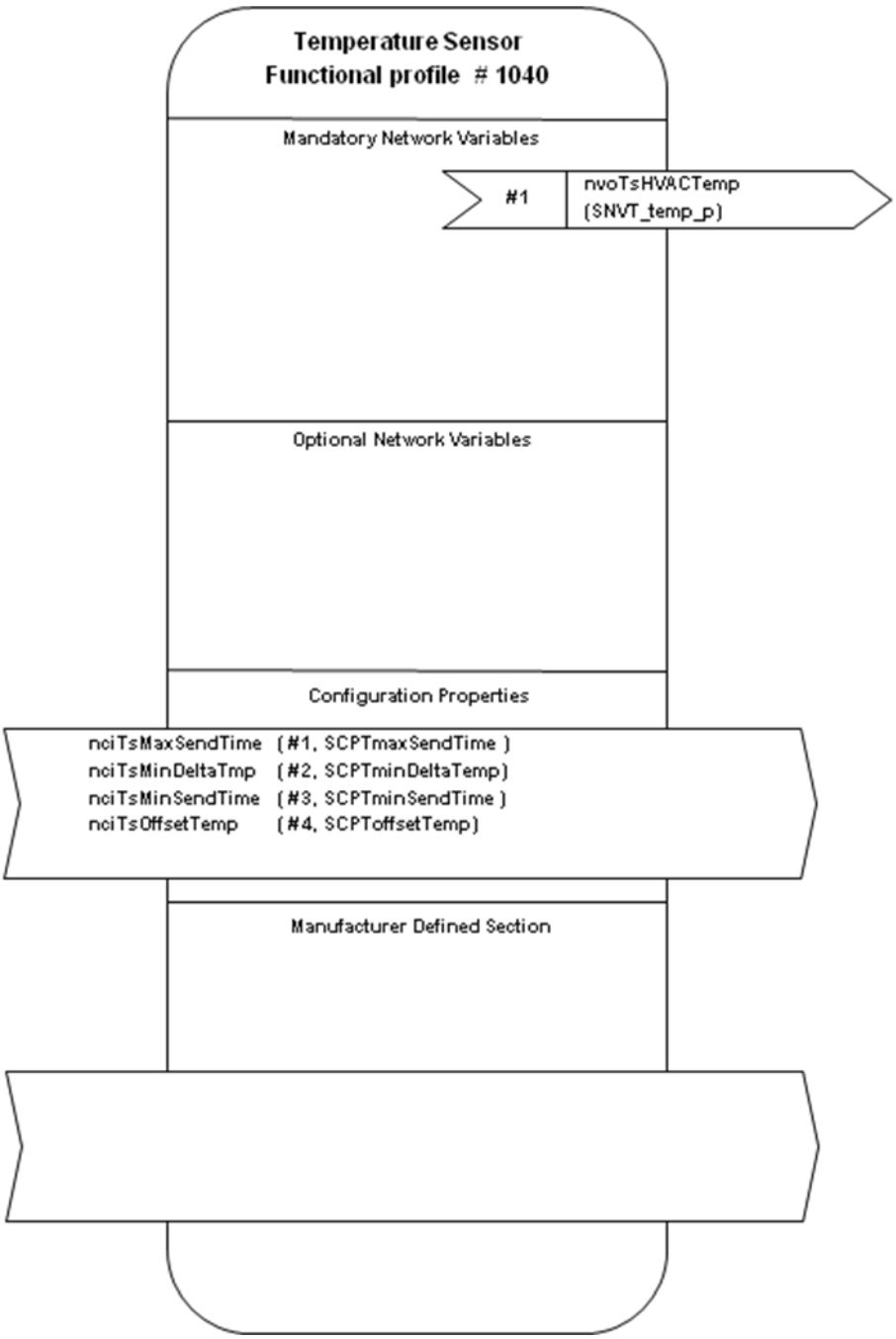
# Contenido

**1. Descripción ..... 3**
**2. Interfaz de red ..... 3**
**3. Variables de red ..... 4**
3.1. nvoTsHVACTemp ..... 4
**4. Variables de configuración ..... 5**
4.1. nciTsMaxSendTime ..... 5
4.2. nciTsMinDeltaTmp ..... 5
4.3. nciTsMinSendTime ..... 5
4.4. nciTsOffsetTemp ..... 6

# 1. Descripción

Este documento describe el perfil de un sensor de temperatura. Este objeto se utiliza en dispositivos que contienen un sensor de temperatura.

# 2. Interfaz de red



# 3. Variables de red

## *3.1. nvoTsHVACTemp*

```
network output SNVT_temp_p nvoTsHVACTemp
```

Esta variable de salida proporciona el valor de temperatura ambiente medido por el sensor.

### *Tipo*

***SNVT_temp_p***

```
typedef signed long SNVT_temp_p;
```

### *Margen de valores*

```
-273,17 .. 327,66 (con 0,01 grados Celsius de
resolución)
```

### *Valor inválido*

```
32767 (0x7FFF)
```

### *Valor por defecto*

Desconocido

# 4. Variables de configuración

## 4.1. nciTsMaxSendTime

```
network input SCPTmaxSendTime cp nciTsmaxSendTime
```

Esta propiedad de configuración se utiliza para controlar el tiempo máximo que puede pasar antes que el objeto envíe automáticamente el valor actual de la variable de salida `nvoTsHVACTemp`. Esto proporciona un periodo de transmisión que asegura que el objeto está activo. Se puede deshabilitar su funcionamiento definiendo su valor a cero.

### *Tipo*

***SCPTmaxSendTime, derivado de SNVT_time_sec.***

### *Margen de valores*

0 .. 6553,4 segundos (0,1 segundos de resolución)

### *Valor por defecto*

{ 3000 } 5 minutos

### *Notas*

Si el valor de la propiedad de configuración es 0, el objeto no actualizará automáticamente el valor de la variable de salida.

## 4.2. nciTsMinSendTime

```
network input SCPTminSendTime cp nciTsMinSendTime
```

Esta propiedad de configuración indica el mínimo periodo de tiempo entre transmisiones consecutivas de la variable de salida `nvoTsHVACTemp`. Esta propiedad permite limitar la cantidad de valores a enviar por unidad mínima de tiempo.

### *Tipo*

***SCPTminSendTime, derivado de SNVT_time_sec***

### *Margen de valores*

0 .. 6553,4 segundos (0,1 segundos de resolución)

### *Valor por defecto*

{ 50 } 5 segundos

### *Notas*

Si el valor de esta propiedad de configuración es 0, el valor de la variable `nvoTsHVACTemp` se envía cada vez que cambia de valor.

## *4.3. nciTsMinDeltaTmp*

```
network input SCPTminDeltaTemp cp nciTsMinDeltaTmp
```

Esta propiedad de configuración se utiliza para definir el valor mínimo que debe variar la salida `nvoTsHVACTemp` para transmitir de nuevo su valor por la red.

### *Tipo*

***SCPTminDeltaTemp, derivado de SNVT_temp_p***

### *Margen de valores*

-273,17 .. 327,66 (resolución de 0,01 grados Celsius)

### *Valor por defecto*

{1 } 1 grado Celsius

## *4.4. nciTsOffsetTemp*

```
network input SCPToffsetTemp cp nciTsOffsetTemp
```

Esta propiedad de configuración se utiliza para ajustar el equipo añadiendo un diferencial de temperatura al valor de `nvoTsHVACTemp`.

### *Tipo*

***SCPToffsetTemp, derivado de SNVT_temp_p.***

## *Margen de valores*

-273,17 .. 327,66 (resolución de 0,01 grados Celsius)

## *Valor por defecto*

{ 0 } 0 grados Celsius

e-controls
electronic intelligent controls, s.l.

## Perfil Funcional:

# Constant Light Controller

## 10501 r1.0

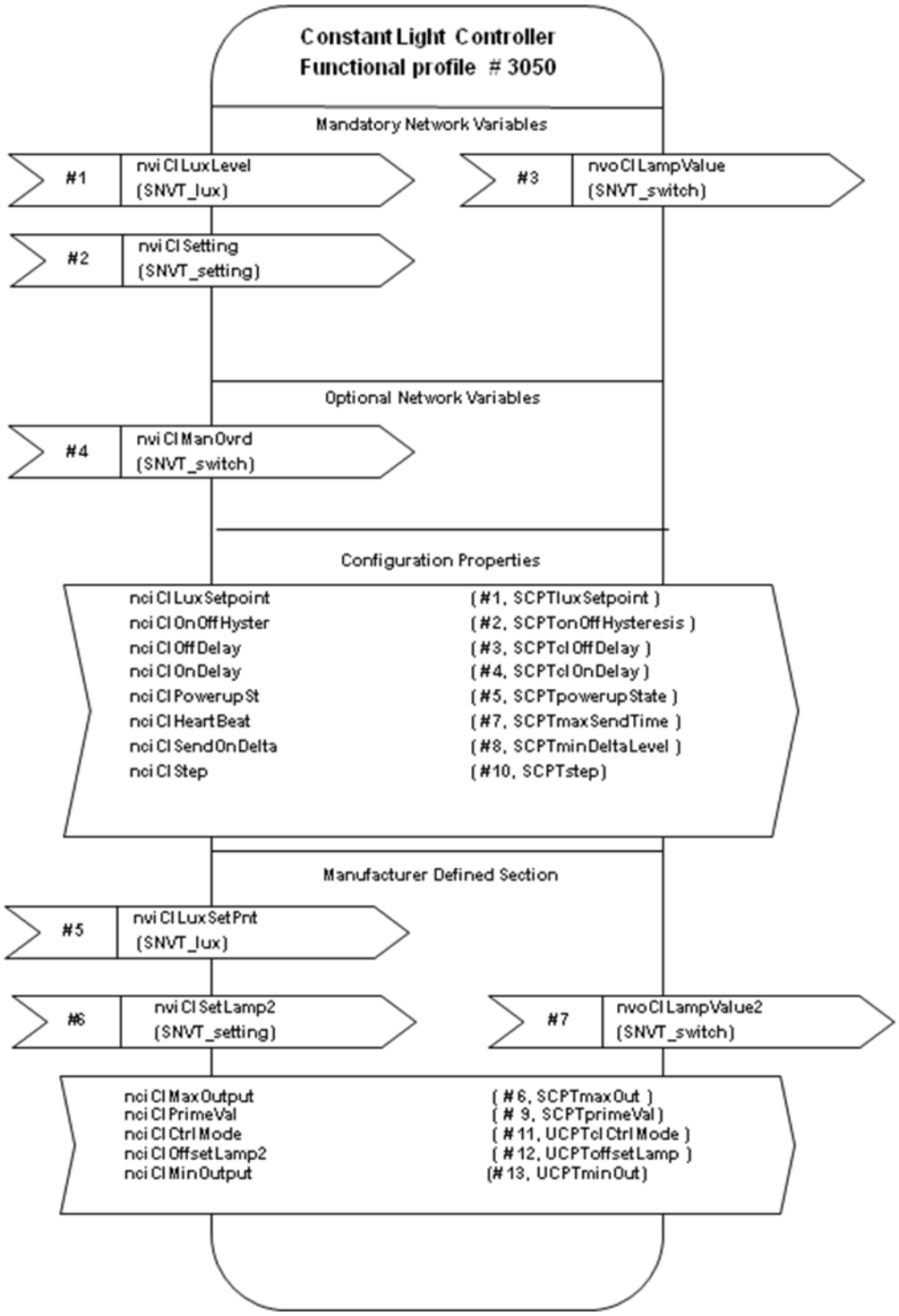
# Contenido

**1. Descripción ..... 3**
**2. Interfaz de red ..... 3**
**3. Variables de red ..... 5**
3.1. nviClLuxLevel ..... 5
3.2. nviClLuxSetPnt ..... 5
3.3. nviClSetting ..... 7
3.4. nviClSetLamp2 ..... 8
3.5. nviClManOvrd ..... 10
3.6. nvoClLampValue ..... 11
3.7. nvoClLampValue2 ..... 11
**4. Variables de configuración ..... 13**
4.1. nciClLuxSetpoint ..... 13
4.2. nciClOnOffHyster ..... 14
4.3. nciClOffDelay ..... 14
4.4. nciClOnDelay ..... 15
4.5. nciClPowerupSt ..... 16
4.6. nciClMaxOutput ..... 17
4.7. nciClMinOutput ..... 17
4.8. nciClHeartBeat ..... 18
4.9. nciClSendOnDelta ..... 18
4.10. nciClPrimeVal ..... 19
4.11. nciClStep ..... 19
4.12. nciClCtrlMode ..... 20
4.13. nciClOffsetLamp2 ..... 20
**5. Histórico de revisiones ..... 22**

# 1. Descripción

El objeto *Constant Light Controller (CLC)* se encarga de proporcionar un nivel de luminosidad constante en una estancia a través de un mecanismo de ajuste automático de la luminosidad de una luminaria o grupo de luminarias. El nivel de iluminación se regulará automáticamente en función de la luz ambiente de la estancia y de la aportación de luz exterior en la misma a partir de un nivel de consigna de luz previamente definido.

# 2. Interfaz de red


Constant Light Controller
Functional profile # 3050
Mandatory Network Variables
#1
nviClLuxLevel
(SNVT_lux)
#3
nvoClLampValue
(SNVT_switch)
#2
nviClSetting
(SNVT_setting)
Optional Network Variables
#4
nviClManOvrd
(SNVT_switch)
Configuration Properties
nciClLuxSetpoint (#1, SCPTluxSetpoint )
nciClOnOffHyster (#2, SCPTonOffHysteresis )
nciClOffDelay (#3, SCPTclOffDelay )
nciClOnDelay (#4, SCPTclOnDelay )
nciClPowerupSt (#5, SCPTpowerupState )
nciClHeartBeat (#7, SCPTmaxSendTime )
nciClSendOnDelta (#8, SCPTminDeltaLevel )
nciClStep (#10, SCPTstep)
Manufacturer Defined Section
#5
nviClLuxSetPnt
(SNVT_lux)
#6
nviClSetLamp2
(SNVT_setting)
#7
nvoClLampValue2
(SNVT_switch)
nciClMaxOutput (#6, SCPTmaxOut )
nciClPrimeVal (# 9, SCPTprimeVal)
nciClCtrlMode (#11, UCPTclCtrlMode )
nciClOffsetLamp2 (#12, UCPToffsetLamp )
nciClMinOutput (#13, UCPTminOut)

# 3. Variables de red

## 3.1. nviClLuxLevel

```
network input SNVT_lux nviClLuxLevel;
```

Esta variable le proporciona al controlador el nivel de luz ambiente de la zona donde está instalado el sensor.

### *Tipo*

***SNVT_lux***

```
typedef unsigned long SNVT_lux;
```

### *Margen de valores*

0 .. 1000 (0 .. 1000 lux, con resolución 1 lux)

### *Valor por defecto*

{ 0 } 0 lux

## 3.2. nviClLuxSetPnt

```
network input SNVT_lux nviClLuxSetPnt;
```

Esta variable de red se utiliza para definir un nuevo valor de consigna para el CLC. La variable se inicializa automáticamente con el valor de `nciClLuxSetPoint` después de un reset.

Cuando se modifica el valor de `nviClLuxSetPnt`, se sobrescribe internamente el valor de la propiedad de configuración `nciClLuxSetPoint`.

Cuando el equipo está configurado cómo un controlador de luz ON/OFF (ver 4.11 `nciClCtrlMode`) esta variable de red se utiliza como umbral de luz del controlador ON/OFF.

### *Tipo*

***SNVT_lux***

```
typedef unsigned long SNVT_lux;
```

### *Margen de valores*

0 .. 1000 (0 .. 1000 lux, 1 lux de resolución)

### *Valor por defecto*

El valor se toma de la propiedad de configuración `nciClLuxSetPoint`.

## 3.3. nviClSetting

```
network input SNVT_setting nviClSetting;
```

Esta variable selecciona el modo de operación del CLC y permite ajustar la consigna de luz del controlador. Los modos de operación son SET_ON, SET_OFF, SET_UP, SET_DOWN y SET_STOP:

- El modo SET_ON activa el CLC, el cual empieza a controlar el valor del nivel de luz nvoClLampValue para que el valor proporcionado por nviClLuxLevel se iguale al valor de consigna nviClLuxSetPnt.
- El modo SET_OFF desactiva el CLC y fija el valor de nvoClLampValue a {OFF,0}.
- La consigna del CLC puede ser modificada temporalmente mediante los modos SET_UP y SET_DOWN, que incrementan y decrementan respectivamente el valor de consigna hasta que se envía el valor SET_STOP. Los cambios así realizados en el valor de la consigna no se guardan permanentemente. La próxima vez que se seleccione el modo SET_ON se restablecerá el valor original de consigna guardado en la variable nviClLuxSetPnt.

### *Tipo*

***SNVT_setting***

```
typedef struct
{
     setting_t function;
     unsigned short setting;
     signed long rotation;
} SNVT_setting;
```

### *Margen de valores*

`function`

| Valor | Descripción | Descripción |
|---|---|---|
| -1 | SET_NUL | Inválido, continuar con modo de operación actual |
| 0 | SET_OFF | Desactivar controlador y salida |
| 1 | SET_ON | Activar controlador y salida |
| 2 | SET_DOWN | Bajar nivel de consigna gradualmente hasta que se reciba SET_STOP o se alcance la |

| | | |
|---|---|---|
| | | consigna mínima permitida. |
| 3 | SET_UP | Subir nivel de consigna gradualmente, hasta que se reciba SET_STOP o se alcance la consigna máxima permitida. |
| 4 | SET_STOP | Finalizar la función de incremento o decremento de consigna. |

`setting`

No se utiliza.

`rotation`

No se utiliza.

### *Valor por defecto*

`{SET_NUL, 0, 0}` -

## *3.4. nviClSetLamp2*

```
network input SNVT_setting nviClSetLamp2;
```

Esta variable se utiliza para controlar el estado de la segunda salida de control de iluminación definida por `nvoClLampValue2`.

### *Tipo*

***SNVT_setting***

```
typedef struct
{
     setting_t function;
     unsigned short setting;
     signed long rotation;
} SNVT_setting;
```

### *Margen de valores*

`function`

| Value | Descripción | Description |
|---|---|---|

| | | |
|---|---|---|
| -1 | SET_NUL | Valor inválido. Mantiene el mismo modo de funcionamiento. |
| 0 | SET_OFF | Desactiva el controlador de la salida nvoClLampValue2. |
| 1 | SET_ON | Activa el controlador de la salida nvoClLampValue2. |

`setting`

No se utiliza.

`rotation`

No se utiliza.

### *Valor por defecto*

```
{SET_NUL, 0, 0}     -
```

## 3.5. nviClManOvrd

```
network input SNVT_switch nviClManOvrd;
```

Esta variable proporciona la posibilidad de controlar manualmente la variables de salida `nvoClLampValue` del controlador.

Cualquier cambio en esta variable provoca que el controlador se desactive y que se copie su valor en la variable de salida `nvoClLampValue`.

Si el valor de `nviClManOver.state` es -1, el controlador se activa de nuevo y la variable de salida `nvoClLampValue` tomará el valor correspondiente.

### *Tipo*

***SNVT_switch***

```
typedef struct
{
     unsigned value;
     signed state;
} SNVT_switch;
```

### *Margen de valores*

`value`

| | |
|---|---|
| 0 .. 200 | ( 0% .. 100%, con resolución 0,5% ) |

`state`

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (Nota: Si `value` = 0, se interpreta como Apagado) |
| -1 | Inválido: Se habilita el funcionamiento del controlador. |

### *Valor por defecto*

| | |
|---|---|
| { 0, -1 } | Controlador habilitado. |

## 3.6. nvoClLampValue

```
network output SNVT_switch nvoClLampValue;
```

Esta variable proporciona el nivel de luminosidad que deben adoptar las luminarias asociadas a este dispositivo.

### Tipo

***SNVT_switch***

```
typedef struct
{
     unsigned value;
     signed state;
} SNVT_switch;
```

### Margen de valores

`value`

| | |
|---|---|
| 0 .. 200 | ( 0% .. 100%, con resolución 0,5% ) |

`state`

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (Nota: Si value = 0, se interpreta como Apagado) |
| -1 | Inválido: El valor no es válido. |

### Valor por defecto

| | |
|---|---|
| { 0, 0 } | Apagado |

## 3.7. nvoClLampValue2

```
network output SNVT_switch nvoClLampValue2;
```

Esta variable permite controlar una luminaria de una zona secundaria. El valor de esta segunda salida es el de la salida principal `nvoClLampValue`, más un offset definido por la propiedad de entrada `nciClOffsetLamp2`.

## *Tipo*

***SNVT_switch***

```
typedef struct
{
     unsigned value;
     signed state;
} SNVT_switch;
```

## *Margen de valores*

`value`

| | |
|---|---|
| 0 .. 200 | ( 0% .. 100%, 0,5% resolución) |

`state`

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (Note: Si `value` = 0, la salida es OFF) |
| -1 | Valor inválido. |

## *Valor por defecto*

| | |
|---|---|
| { 0, 0 } | Salida es Apagado |

# 4. Variables de configuración

## *4.1. nciClLuxSetpoint*

```
network input SCPTluxSetpoint cp nciClLuxSetpoint;
```

Esta propiedad de configuración establece la consigna de luminosidad que se desea en la zona. El valor de esta propiedad también se modifica cuando se cambia el valor de la variable de red `nviClLuxSetPnt`.

El valor interno de consigna puede ser modificado temporalmente a través de la variable `nviClSetting` sin que afecte al parámetro `nciClLuxSetpoint`.

Cuando el equipo está configurado cómo un Controlador ON/OFF (ver 4.12 `nciClCtrlMode`) esta propiedad actúa cómo el umbral de activación del Controlador ON/OFF.

### *Tipo*

***SCPTluxSetpoint, derivado de SNVT_lux.***

```
typedef unsigned long SNVT_lux;
```

### *Margen de valores*

0 .. 1000 (0 .. 1000 lux, con resolución 1 lux)

### *Valor por defecto*

{ 500 } 500 lux

## 4.2. nciClOnOffHyster

```
network input SCPTonOffHysteresis cp nciClOnOffHyster;
```

Porcentaje de consigna necesario para activar y desactivar automáticamente la iluminación a través de la variable de salida `nvoClLampValue`. En controlador actúa sobre la salida según el valor de esta variable y de las variables `nciClOffDelay` y `nciClOnDelay` (ver 4.3 y 4.4).

Esta variable no se utiliza cuando el equipo está configurado en modo ON/OFF.

### *Tipo*

***SCPTonOffHysteresis, derivado de SNVT_level_cont.***

```
typedef unsigned short SNVT_level_cont;
```

### *Margen de valores*

| | |
|---|---|
| 0 .. 200 | (0% .. 100%, con resolución 0,5%) |

### *Valor por defecto*

| | |
|---|---|
| { 0 } | 0%, en modo de trabajo CLC, desactiva el apagado automático. |

## 4.3. nciClOffDelay

```
network input SCPTclOffDelay cp nciClOffDelay;
```

Esta variable define el tiempo necesario durante el que se debe cumplir la condición de valor de consigna + histéresis (definido por la propiedad `nciOnOffHyster`) respecto al nivel de iluminación en la zona, para poder apagar la iluminación.

En modo CLC esta propiedad entra en funcionamiento cuando el nivel de iluminación esta al mínimo.

### *Tipo*

***SCPTclOffDelay, derivado de SNVT_time_sec.***

### *Margen de valores*

0 – 6553,4 s (0,1 segundos de resolución)

### *Valor por defecto*

{ 3000 } 5 minutos.

## *4.4. nciClOnDelay*

```
network input SCPTclOnDelay cp nciClOnDelay;
```

Esta variable define el tiempo necesario durante el que se debe cumplir la condición de valor de consigna - histéresis (definido por la propiedad `nciOnOffHyster`) respecto al nivel de iluminación en la zona, para poder encender la iluminación.

### *Tipo*

***SCPTclOffDelay, derivado de SNVT_time_sec.***

### *Margen de valores*

0 – 6553,4 s (0,1 segundos de resolución)

### *Valor por defecto*

{ 50 } 5 segundos.

## 4.5. nciClPowerupSt

```
network input SCPTpowerupState nciClPowerupSt;
```

Esta variable establece el estado del controlador CLC y de la salida `nvoCLLampValue` después de un reset.

### Tipo

***SNVT_setting***

```
typedef struct
{
     setting_t function;
     unsigned short setting;
     signed long rotation;
} SNVT_setting;
```

### Margen de valores

`function`

| Valor | Descripción | Descripción |
|---|---|---|
| `0` | `SET_OFF` | Desactivar controlador y salida `nvoCLLampValue` |
| `1` | `SET_ON` | Activar controlador y salida `nvoCLLampValue` |
| `Otro` | `-` | Desactivar controlador y salida `nvoCLLampValue` |

`setting`

0 No se utiliza.

`rotation`

0 No se utiliza.

### Valor por defecto

`{SET_OFF, 0, 0}` Controlador desactivado.

## 4.6. nciClMaxOutput

```
network input SCPTmaxOut cp nciClMaxOutput;
```

Esta propiedad de configuración fija el valor máximo que puede adoptar la variable de salida `nvoClLampValue` cuando el controlador está configurado en modo CLC.

Esta propiedad no limita el valor de la salida `nvoClLampValue` cuando se actúa a través de la variable `nviClManOvrd`.

### *Tipo*

`SCPTmaxOut`***, derivado de SNVT_lev_cont.***

### *Margen de valores*

0,0 % – 100,0 % (resolución de incrementos del 0,5%)

### *Valor por defecto*

{ 200 } 100 %

## 4.7. nciClMinOutput

```
network input UCPTminOut cp nciClMinOutput;
```

Esta propiedad de configuración fija el valor mínimo que puede adoptar la variable de salida `nvoClLampValue` cuando el controlador está configurado en modo CLC.

Esta propiedad no limita el valor de la salida `nvoClLampValue` cuando se actúa a través de la variable `nviClManOvrd`.

### *Tipo*

`UCPTminOut`***, derivado de SNVT_lev_cont.***

### *Margen de valores*

0,0 % – 100,0 % (0,5% steps resolution)

### *Valor por defecto*

{ 20 } 10 %

## 4.8. nciClHeartBeat

```
network input SCPTmaxSendTime cp nciClHeartBeat;
```

Esta propiedad de configuración se utiliza para indicar el periodo máximo de tiempo que debe transcurrir antes de que el equipo transmita de nuevo automáticamente el valor de `nvoClLampValue`. Esta propiedad asegura que el objeto está activo. Se puede deshabilitar asignándole el valor cero.

### *Tipo*

***SCPTmaxSendTime, derivado de SNVT_time_sec.***

### *Margen de valores*

0 – 6553,4 (0.1 segundos de resolución)

### *Valor por defecto*

{ 3000 } 5 minutos

## 4.9. nciClSendOnDelta

```
network input SCPTminDeltaLevel cp nciClSendOnDelta;
```

Esta propiedad de configuración define el valor de incremento que se debe producir en la variable `nvoClLampValue` para enviar su valor por la red. El valor de la variable siempre se transmite cuando pasa a estado ON o OFF.

### *Tipo*

***SCPTminDeltaLevel, derivado de SNVT_lev_cont.***

### *Margen de valores*

0,0 % – 100,0 % (resolución de incrementos del 0,5%)

### *Valor por defecto*

{ 1 } 0,5 %

## 4.10. nciClPrimeVal

```
network input SCPTprimeVal cp nciClPrimeVal;
```

Esta propiedad de configuración define el valor que debe tomar la variable de salida `nvoClLampValue` después de un reset o cada vez que se activa de nuevo el controlador CLC.

### *Tipo*

***SCPTprimeVal, derivado de SNVT_switch.***

```
typedef struct
{
        unsigned value;
        signed state;
} SNVT_switch;
```

### *Margen de valores*

Value

| | |
|---|---|
| 0 ..200 | (0%... 100%, resolución 0,5%) |

State

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (si value = 0 la salida es OFF) |
| -1 | Inválido |

### *Valor por defecto*

| | |
|---|---|
| {1, 100 } | ON, 50%. |

## 4.11. nciClStep

```
network input SCPTstep cp nciClStep;
```

Esta propiedad de configuración se utiliza en el controlador CLC para definir el incremento del valor a aplicar en la variable de salida `nvoClLampValue` hasta ajustar la iluminación al valor de consigna configurado.

### *Tipo*

***SCPTstep, derivado de SNVT_lev_cont.***

### *Margen de valores*

0,0 % – 100,0 % (resolución en incrementos del 0,5%)

### *Valor por defecto*

{ 4 } 2 %

## *4.12. nciClCtrlMode*

```
network input UCPTclCtrlMode cp nciClCtrlMode;
```

Esta propiedad de configuración determina el modo de funcionamiento del controlador. El controlador puede trabajar en modo constant light controller (CLC) o en modo ON/OFF.

### *Tipo*

`UCPTclCtrlMode`***, derivado de UNVT_cl_Ctrl_Mode.***

```
typedef enum
{
        CL_ONOFF = 0,
        CL_CONSTANT = 1
} clc_mode_t;
```

### *Margen de valores*

| | |
|---|---|
| 0 | Modo ON/OFF |
| 1 | Modo Constant Light Controller |

### *Valor por defecto*

{ 1 } Modo Constant Light Controller

## *4.13. nciClOffsetLamp2*

```
network input UCPToffsetLamp cp nciClOffsetLamp2;
```

Esta propiedad de configuración se utiliza en la variable de salida `nvoClLampValue2` para determinar el incremento de nivel de luz respecto al valor de la variable de salida `nvoClLampValue`.

### *Tipo*

***UCPToffsetLamp, derivado de SNVT_lev_cont.***

### *Margen de valores*

0,0 % – 100,0 % (resolución en incrementos del 0,5%)

### *Valor por defecto*

{40} 20 %

e-controls
electronic intelligent controls, s.l.

## Perfil Funcional:

# Occupancy Controller

## 10501 r1.0

# Contenido

**1. Descripción ..... 3**
1.1. Funcionamiento ..... 3
**2. Interfaz de red ..... 4**
**3. Variables de red ..... 5**
3.1. nviOcOccupancy ..... 5
3.2. nviOcSetting ..... 6
3.3. nviOcManOverride ..... 7
3.4. nviOcSecondary ..... 8
3.5. nvoOcLampValue ..... 8
3.6. nvoOcSetting ..... 9
**4. Variables de configuración ..... 11**
4.1. nciOcHoldTime ..... 11
4.2. nciOcHeartBeat ..... 11
4.3. nciOcPrimeVal ..... 12
4.4. nciOcSecondVal ..... 13
4.5. nciOcUnocVal ..... 13

# 1. Descripción

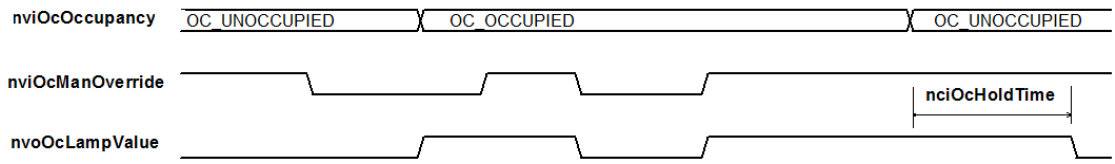
El objeto *Occupancy Controller* permite controlar un actuador, normalmente una luminaria, en función del estado de ocupación de una estancia. Cuando la estancia está ocupada se activa la salida del controlador, mientras que si la estancia pasa a estar desocupada la salida se desactiva.

Dispone de una entrada para controlar manualmente la salida cuando la estancia está ocupada.

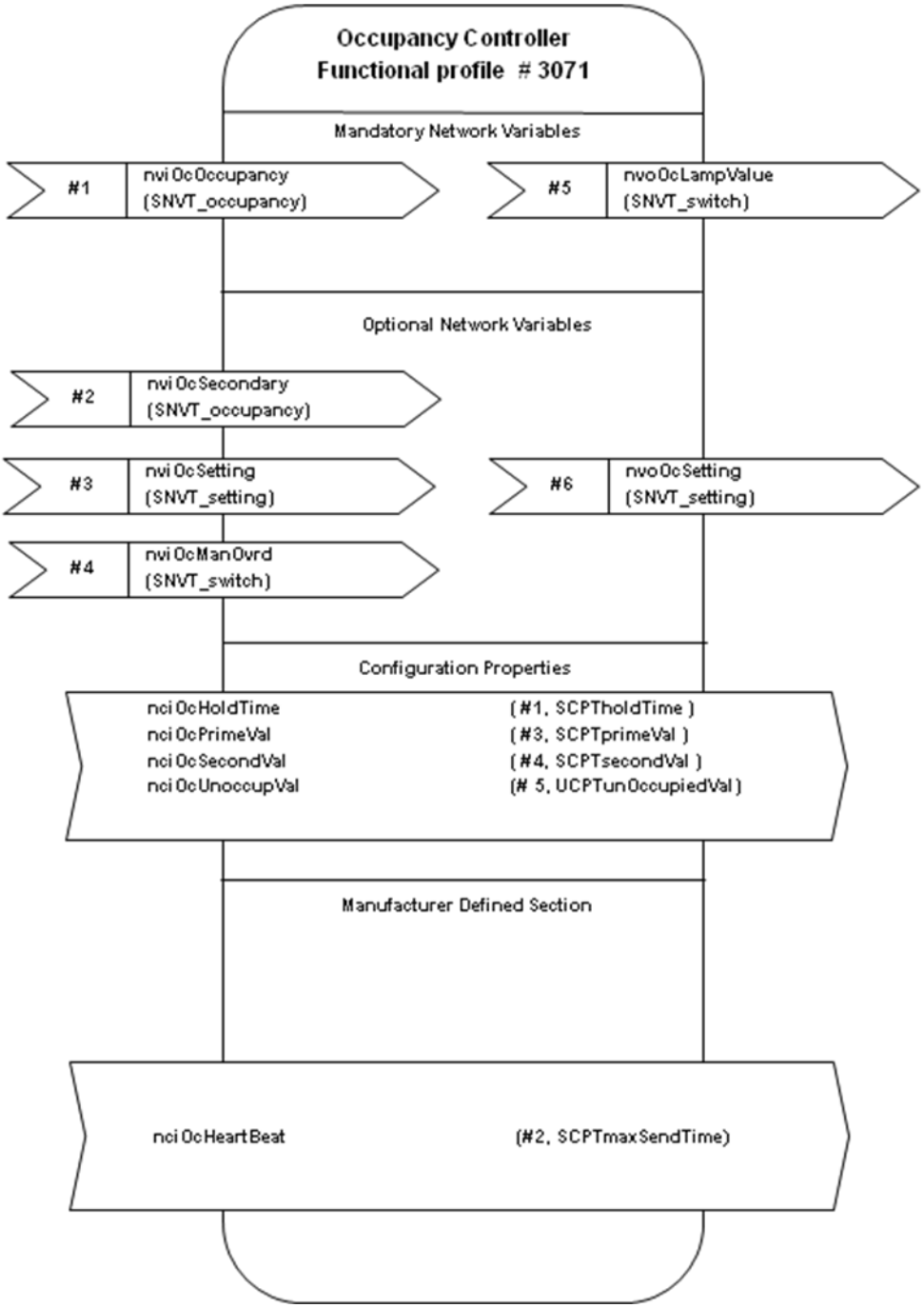
## *1.1. Funcionamiento*

El valor de la variable de salida `nvoOcLampValue` viene determinado por el valor de `nviOcOccupancy`:

- Cuando pasa de Desocupado a Ocupado, la salida se activa automáticamente.
- Cuando pasa de Ocupado a Desocupado la salida se desactiva automáticamente pasado el tiempo definido en `nciOcHoldTime`.
- Mientras esté en Ocupado, se puede cambiar el valor de la salida mediante la variable `nviOcManOverride`.



# 2. Interfaz de red



# 3. Variables de red

## *3.1. nviOcOccupancy*

```
network input SNVT_occupancy nviOcOccupancy;
```

Esta variable proporciona al controlador el estado de ocupación de la zona donde se encuentra instalado el equipo.

### *Tipo*

***SNVT_occupancy***

```
typedef occup_t SNVT_occupancy;
```

### *Margen de valores*

| Valor | Descripción | Descripción |
|---|---|---|
| 0 | OC_OCCUPIED | Estado Ocupado |
| 1 | OC_UNOCCUPIED | Estado Desocupado |
| 2 | OC_BYPASS | No utilizado |
| 3 | OC_STANDBY | No utilizado |
| -1 | OC_NUL | Valor inválido |

### *Valor por defecto*

{ -1 }  OC_NUL

## 3.2. nviOcSetting

```
network input SNVT_setting nviOcSetting;
```

Esta variable activa o desactiva el controlador. El modo ON activa el controlador y empieza a controlar el valor de `nvoOcLampValue` según el estado de ocupación de la estancia. El modo OFF desactiva el controlador y la salida `nvoOcLampValue`.

### *Tipo*

***SNVT_setting***

```
typedef struct
{
     setting_t function;
     unsigned short setting;
     signed long rotation;
} SNVT_setting;
```

### *Margen de valores*

`function`

| Valor | Descripción | Descripción |
|---|---|---|
| 0 | SET_OFF | Desactivar controlador y salida `nvoOcLampValue` |
| 1 | SET_ON | Activar controlador |

El resto de valores de la enumeración no se utiliza.

`setting`

No se utiliza.

`rotation`

No se utiliza.

### *Valor por defecto*

```
{SET_ON, 0, 0}
```

## 3.3. nviOcManOverride

```
network input SNVT_switch nviOcManOverride;
```

Esta variable fuerza el valor de la salida `nvoOcLampValue` al valor definido en esta variable.

Cuando la variable `nviOcOccupancy` tiene el valor `OC_OCCUPIED`, cualquier cambio en la variable `nviOcManOverride` provoca que el controlador envíe el valor de esta variable a la salida `nvoOcLampValue`.

Cuando la variable `nviOcOccupancy` recibe el valor `OC_UNOCCUPIED`, el controlador fuerza la salida al estado Apagado y los cambios en esta variable quedan sin efecto.

### *Tipo*

***SNVT_switch***

```
typedef struct
{
     unsigned value;
     signed state;
} SNVT_switch;
```

### *Margen de valores*

`value`

| | |
|---|---|
| 0 .. 200 | ( 0% .. 100%, con resolución 0,5% ) |

`state`

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (Nota: Si `value` = 0, se interpreta como Apagado) |
| -1 | Inválido: El valor leído no es válido. |

### *Valor por defecto*

| | |
|---|---|
| { 0, -1 } | Inválido |

## 3.4. nviOcSecondary

```
network input SNVT_occupancy nviOcSecondary;
```

Esta variable se utiliza para definir un segundo nivel de iluminación en una zona. Su activación cargará el valor de `nciOcSecondVal` en la variable de salida `nvoOcLampValue`, siempre y cuando el valor de `nviOcOccupancy` esté desocupado.

### *Tipo*

***SNVT_occupancy***

```
typedef occup_t SNVT_occupancy;
```

### *Margen de valores*

| Valor | Descripción | Descripción |
|---|---|---|
| 0 | OC_OCCUPIED | Estado Ocupado |
| 1 | OC_UNOCCUPIED | Estado Desocupado |
| -1 | OC_NUL | Valor inválido |

### *Valor por defecto*

```
{-1}            OC_NUL
```

## 3.5. nvoOcLampValue

```
network output SNVT_switch nvoOcLampValue;
```

Esta variable se utiliza para controlar un actuador externo. Su activación depende del valor del resto de variables del objeto y su desactivación se puede retardar con el parámetro de configuración `nciOcHoldTime`.

### *Tipo*

***SNVT_switch***

```
typedef struct
{
     unsigned value;
     signed state;
} SNVT_switch;
```

### *Margen de valores*

`value`

| | |
|---|---|
| 0 .. 200 | ( 0% .. 100%, con resolución 0,5% ) |

`state`

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (Nota: Si value = 0, se interpreta como Apagado) |
| -1 | Inválido: El valor no es válido. |

### *Valor por defecto*

| | |
|---|---|
| { 0, 0 } | Apagado |

## *3.6. nvoOcSetting*

```
network input SNVT_setting nvoOcSetting;
```

Esta variable informa del modo de funcionamiento en el que se encuentra en controlador, según la siguiente tabla:

| nviSetting | nviOccupancy | nvoSetting |
|---|---|---|
| SET_ON | OC_OCCUPIED | SET_ON |
| | OC_UNOCCUPIED | SET_OFF |
| SET_OFF | cualquiera | SET_OFF |

Su desactivación se puede retardar con el parámetro de configuración `nciOcHoldTime`.

### *Tipo*

***SNVT_setting***

```
typedef struct
{
     setting_t function;
     unsigned short setting;
     signed long rotation;
} SNVT_setting;
```

### *Margen de valores*

`function`

| Valor | Descripción | Descripción |
|---|---|---|
| 0 | SET_OFF | El controlador y la salida están desactivados |
| 1 | SET_ON | El controlador está activado |

El resto de valores de la enumeración no se utiliza.

`setting`

No se utiliza.

`rotation`

No se utiliza.

### *Valor por defecto*

```
{SET_OFF, 0, 0}
```

# 4. Variables de configuración

## 4.1. nciOcHoldTime

```
network input SCPTholdTime cp nciOcHoldTime;
```

Esta propiedad de configuración establece el periodo de tiempo que debe pasar desde que el controlador pasa a estado Desocupado hasta desactivar automáticamente la salida `nvoOcLampValue`. El valor de este parámetro también afecta a la variable de salida `nvoOcSetting`.

### *Tipo*

***SCPTholdTime, derivado de SNVT_time_sec.***

### *Margen de valores*

| | |
|---|---|
| 0 .. 6553,4 | (0 .. 6553 segundos, con resolución 1 décima de segundo) |

### *Valor por defecto*

| | |
|---|---|
| { 600 } | 60 segundos. |

### *Notas*

Un valor de 0 segundos implica que las variables `nvoOcLampValue` y `nvoOcSetting` se desactiven inmediatamente al pasar el controlador a estado Desocupado.

## 4.2. nciOcHeartBeat

```
network input SCPTmaxSendTime cp nciOcHeartBeat;
```

Esta propiedad de configuración se utiliza para indicar el periodo máximo de tiempo que debe transcurrir antes de que el equipo transmita de nuevo automáticamente los valores de `nvoOcLampValue` y `nvoOcSetting`. Esta propiedad asegura que el objeto está activo. Se puede deshabilitar asignándole el valor cero.

### *Tipo*

***SCPTmaxSendTime, derivado de SNVT_time_sec.***

### *Margen de valores*

0 .. 6553,4 en incrementos de 0,1 s

### *Valor por defecto*

{ 1200 } 2 minutos

## *4.3. nciOcPrimeVal*

```
network input SCPTprimeVal cp nciOcPrimeVal;
```

Esta propiedad de configuración se utiliza para actualizar el valor de la variable `nvoOcLampValue` cuando el controlador pasa a estado Ocupado a través de la variable `nviOcOccupancy`.

### *Tipo*

***SCPTprimeVal, derivado de SNVT_switch.***

```
typedef struct
{
        unsigned value;
        signed state;
} SNVT_switch;
```

### *Margen de valores*

Value

0 .. 200 (0%... 100%, resolución 0,5%)

State

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (si value = 0 es OFF) |
| -1 | Inválido |

### *Valor por defecto*

{1, 200 } ON, 100%.

## 4.4. nciOcSecondVal

```
network input SCPTsecondVal cp nciOcSecondVal;
```

Esta propiedad de configuración se utiliza para actualizar el valor de la variable `nvoOcLampValue` cuando el controlador pasa a estado Ocupado a través de la variable `nviOcSecondary` y la variable `nviOcOccupancy` se encuentre en estado Desocupado.

### *Tipo*

***SCPTsecondVal, derivado de SNVT_switch.***

```
typedef struct
{
        unsigned value;
        signed state;
} SNVT_switch;
```

### *Margen de valores*

Value

| | |
|---|---|
| 0 ..200 | (0%... 100%, resolución 0,5%) |

State

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (si value = 0 es OFF) |
| -1 | Inválido |

### *Valor por defecto*

| | |
|---|---|
| {0, 0 } | OFF – 0%. |

## 4.5. nciOcUnocVal

```
network input UCPTunOccupiedValcp nciOcUnocVal;
```

Esta propiedad de configuración se utiliza para actualizar la variable de salida `nvoOcLampValue` cuando el controlador pasa a estado Desocupado.

### *Tipo*

***UCPTunOccupiedVal, derivado de SNVT_switch.***

```
typedef struct
{
        unsigned value;
        signed state;
} SNVT_switch;
```

## *Margen de valores*

Value

| | |
|---|---|
| 0 ..200 | (0%... 100%, resolución 0,5%) |

State

| | |
|---|---|
| 0 | Apagado |
| 1 | Encendido (si value = 0 es OFF) |
| -1 | Inválido |

## *Valor por defecto*

| | |
|---|---|
| {0, 0 } | OFF, 0%. |