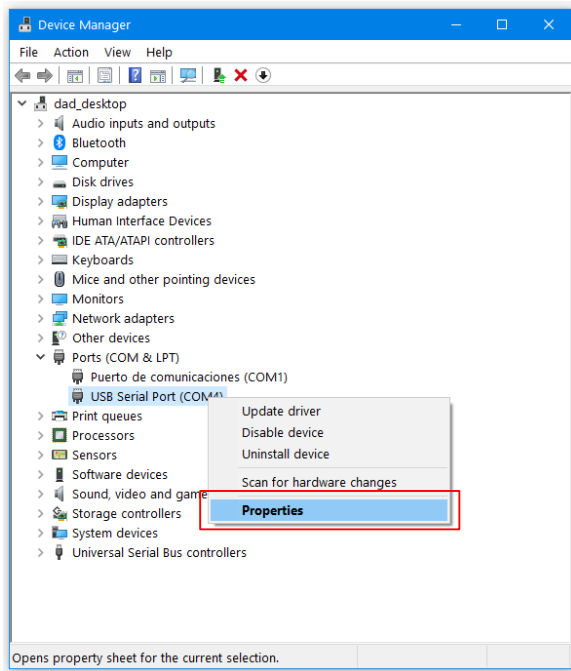
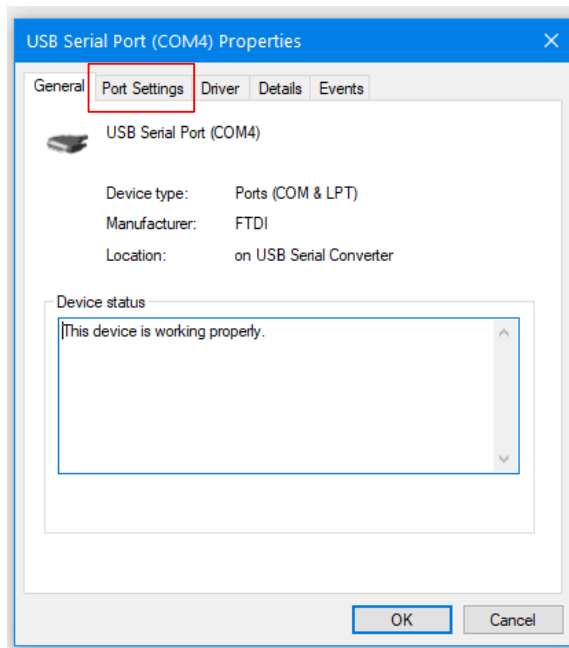


ftdi driver setup

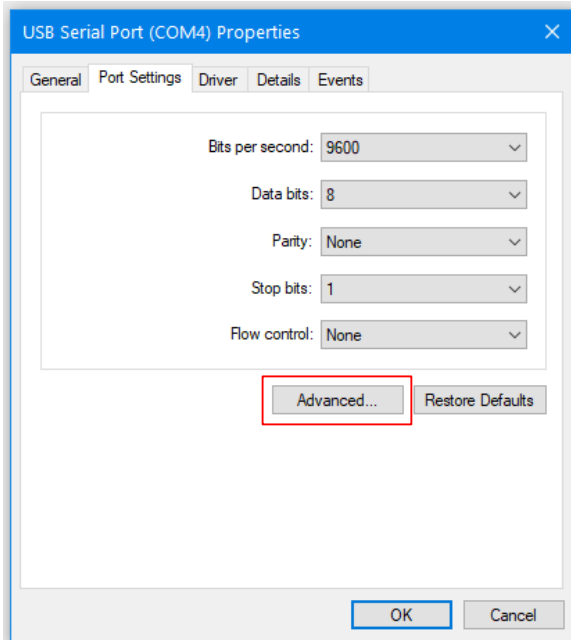
1



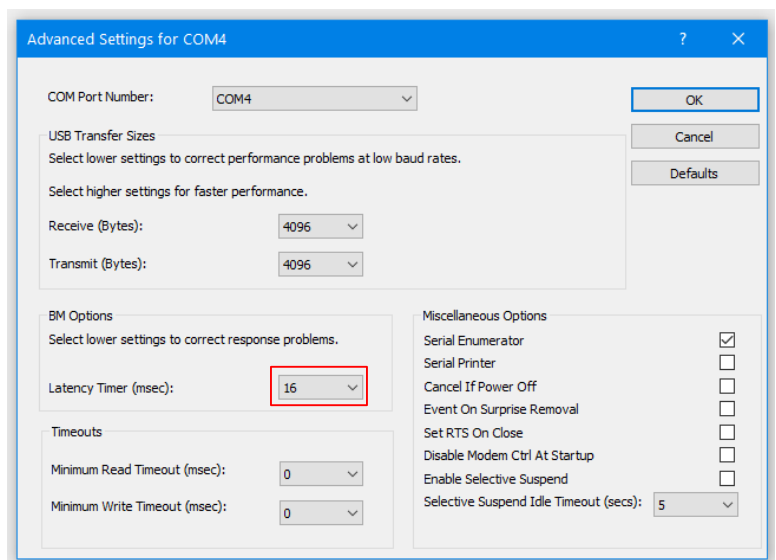
2



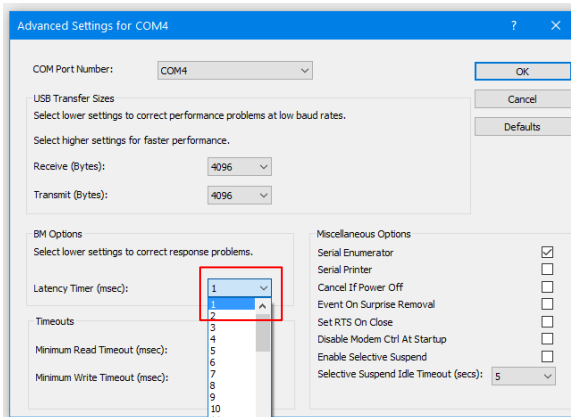
3



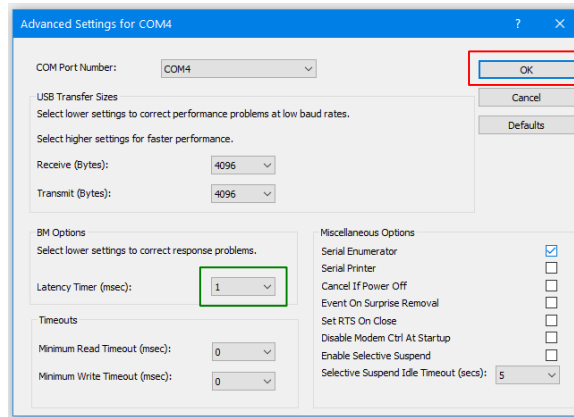
4

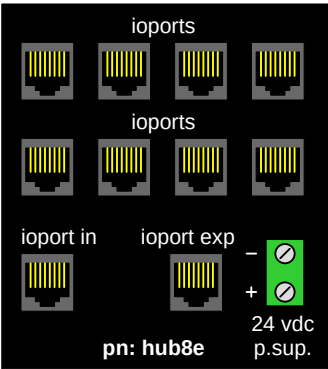
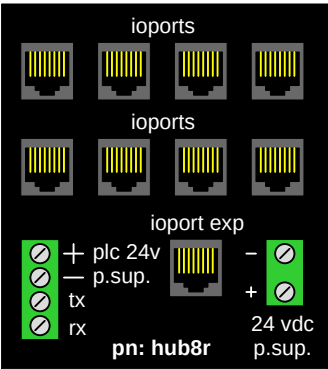
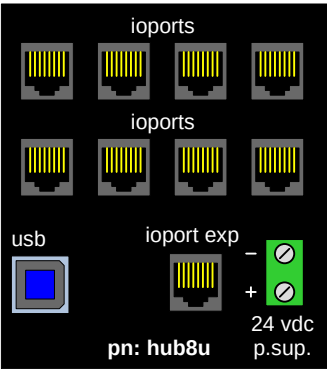


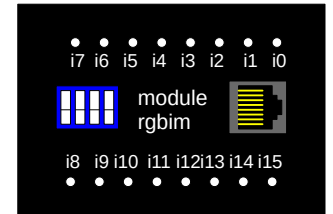
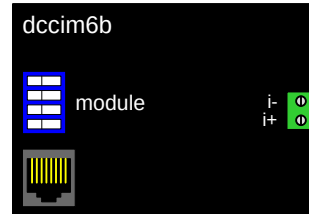
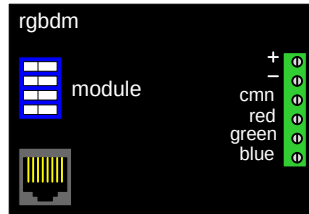
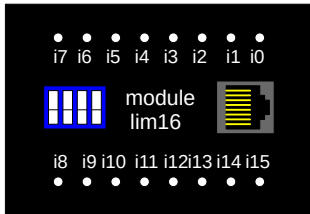
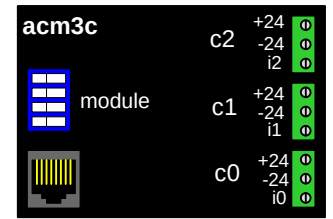
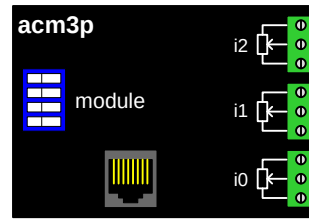
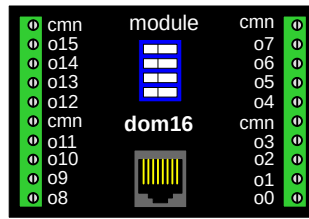
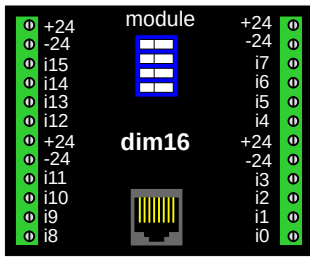
5



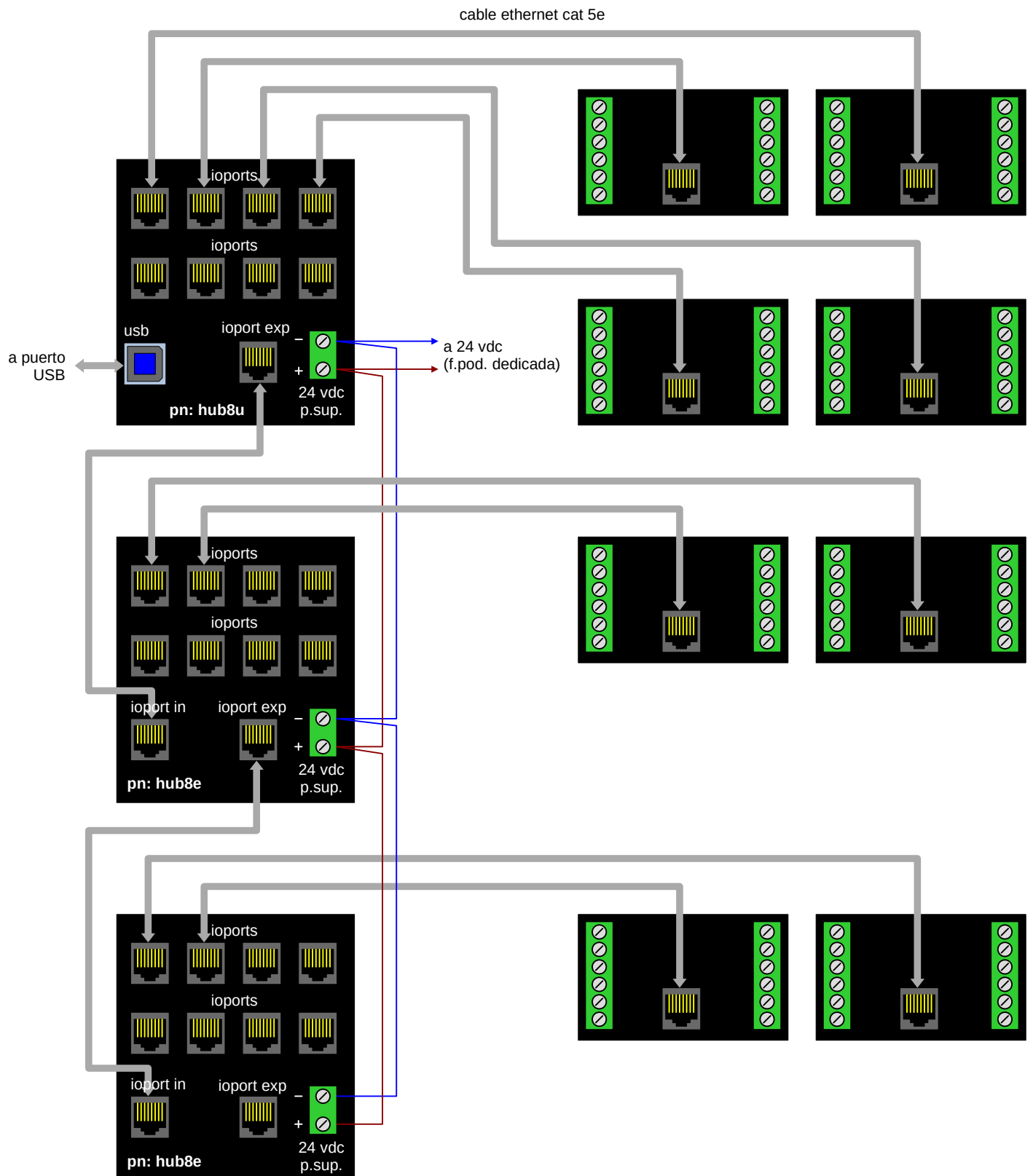
6



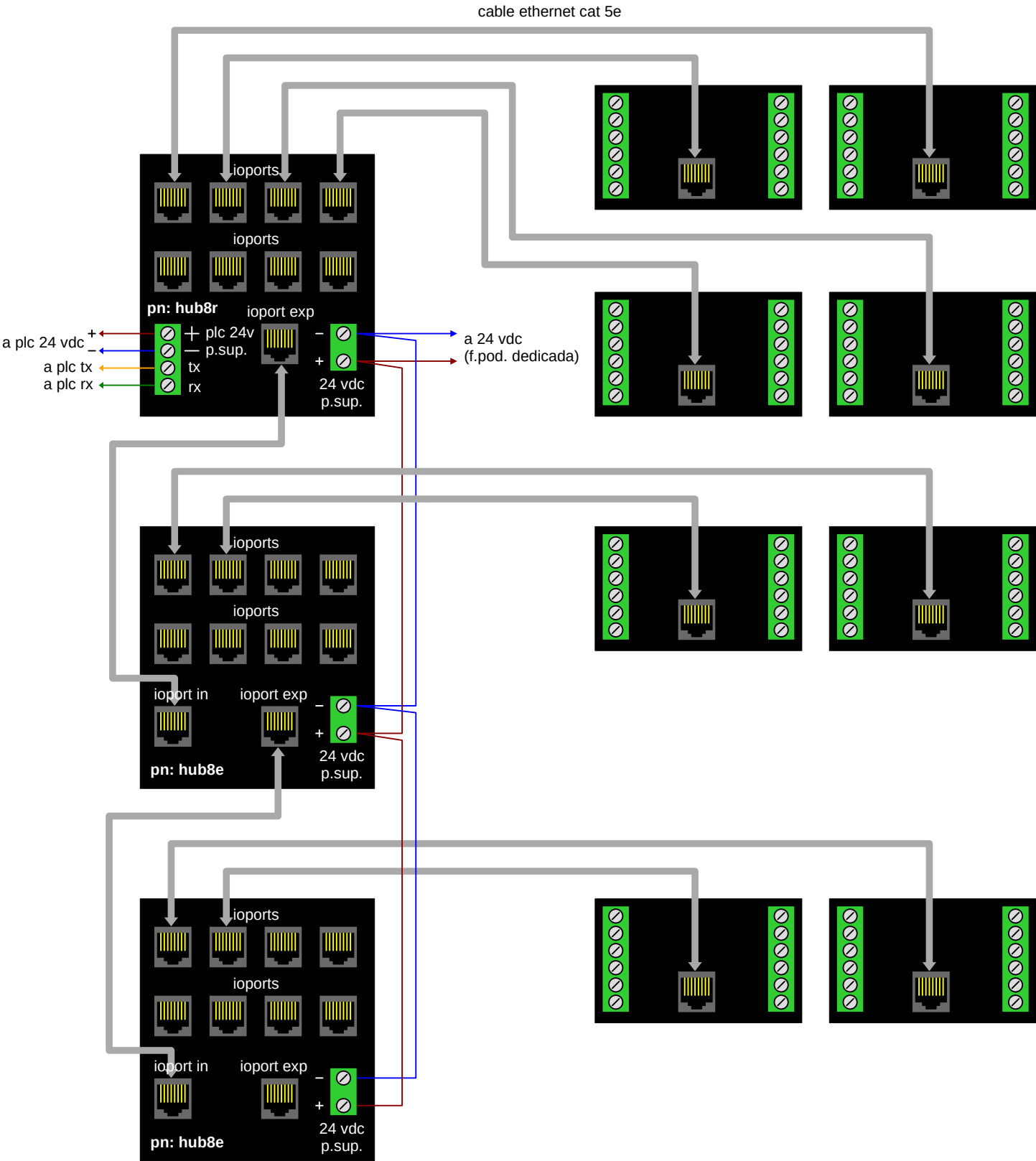




Cableado ioports (con PC)



Cableado ioports (con PLC)



ejemplo

**Especificaciones:**

Voltaje: 24 vdc

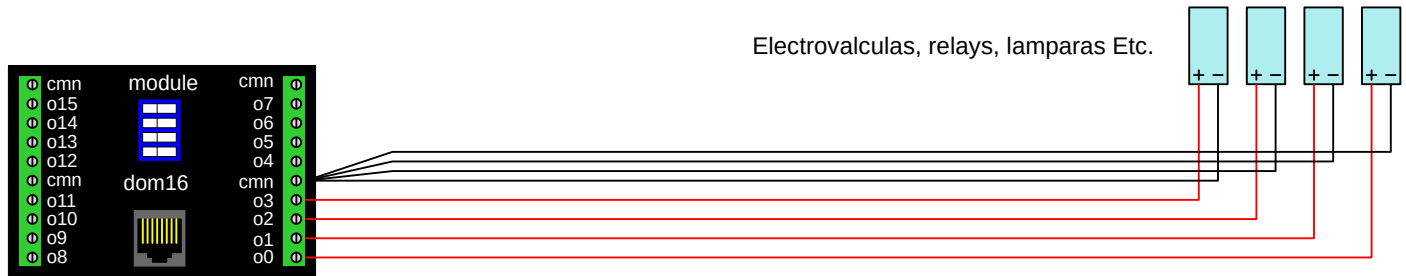
Tipo: PNP

Cantidad máxima de módulos: 8

Comunicación:**Enviar 3 caracteres:**El primer carácter representa el tipo de módulo, en este caso sería una **a**El segundo carácter representa el número de módulo, el cual sería un valor entre **0** y **7**El tercer carácter representa el terminador, el cual sería un carriage return (**<CR>**)**Respuesta:**4 caracteres hexadecimales terminados con un carriage return (**<CR>**), donde:

Los primeros 4 caracteres representan el valor de las 16 entradas, (0000 a FFFF)

El último carácter representa el terminador (**<CR>**)**Ejemplos****Leer el valor de las 16 entradas de un módulo configurado como módulo **0****Enviar: **a0<CR>**Respuesta: **0000<CR>**Los primeros 4 caracteres **0000**, en decimal sería un valor de **0**El último carácter representa el terminador (**<CR>**)**Leer el valor de voltaje de un módulo configurado como módulo **1****Enviar: **a1<CR>**Respuesta: **8000<CR>**Los primeros 4 caracteres **8000**, en decimal sería un valor de **32768**El último carácter representa el terminador (**<CR>**)Para determinar el valor de i0, si el valor decimal **AND 1 = 1**, entonces i0 = 1, de lo contrario i0 = 0Para determinar el valor de i1, si el valor decimal **AND 2 = 2**, entonces i1 = 1, de lo contrario i1 = 0Para determinar el valor de i2, si el valor decimal **AND 4 = 4**, entonces i2 = 1, de lo contrario i2 = 0Para determinar el valor de i3, si el valor decimal **AND 8 = 8**, entonces i3 = 1, de lo contrario i3 = 0Para determinar el valor de i4, si el valor decimal **AND 16 = 16**, entonces i4 = 1, de lo contrario i4 = 0Para determinar el valor de i5, si el valor decimal **AND 32 = 32**, entonces i5 = 1, de lo contrario i5 = 0Para determinar el valor de i6, si el valor decimal **AND 64 = 64**, entonces i6 = 1, de lo contrario i6 = 0Para determinar el valor de i7, si el valor decimal **AND 128 = 128**, entonces i7 = 1, de lo contrario i7 = 0Para determinar el valor de i8, si el valor decimal **AND 256 = 256**, entonces i8 = 1, de lo contrario i8 = 0Para determinar el valor de i9, si el valor decimal **AND 512 = 512**, entonces i9 = 1, de lo contrario i9 = 0Para determinar el valor de i10, si el valor decimal **AND 1024 = 1024**, entonces i10 = 1, de lo contrario i10 = 0Para determinar el valor de i11, si el valor decimal **AND 2048 = 2048**, entonces i11 = 1, de lo contrario i11 = 0Para determinar el valor de i12, si el valor decimal **AND 4096 = 4096**, entonces i12 = 1, de lo contrario i12 = 0Para determinar el valor de i13, si el valor decimal **AND 8192 = 8192**, entonces i13 = 1, de lo contrario i13 = 0Para determinar el valor de i14, si el valor decimal **AND 16384 = 16384**, entonces i14 = 1, de lo contrario i14 = 0Para determinar el valor de i15, si el valor decimal **AND 32768 = 32768**, entonces i15 = 1, de lo contrario i15 = 0

**Especificaciones:**

Voltaje: 24 vdc

Corriente máxima por salida: 250 mA

Tipo: PNP

Cantidad máxima de módulos: 8

Comunicación:**Enviar 7 caracteres:**

El primer carácter representa el tipo de modulo, en este caso seria una **b**
 El segundo carácter representa el numero de modulo, el cual seria un valor entre **0** y **7**
 El tercer carácter representa el valor de las salidas o12 a o15 en hexadecimal
 El cuarto carácter representa el valor de las salidas o8 a o11 en hexadecimal
 El quinto carácter representa el valor de las salidas o4 a o7 en hexadecimal
 El sexto carácter representa el valor de las salidas o0 a o3 en hexadecimal
 El ultimo carácter representa el terminador, el cual seria un carriage return (**<CR>**)

o15	o14	o13	o12	Car 3	o11	o10	o9	o8	Car 4	o7	o6	o5	o4	Car 5	o3	o2	o1	o0	Car 6
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1
0	0	1	0	2	0	0	1	0	2	0	0	1	0	2	0	0	1	0	2
0	0	1	1	3	0	0	1	1	3	0	0	1	1	3	0	0	1	1	3
0	1	0	0	4	0	1	0	0	4	0	1	0	0	4	0	1	0	0	4
0	1	0	1	5	0	1	0	1	5	0	1	0	1	5	0	1	0	1	5
0	1	1	0	6	0	1	1	0	6	0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	0	1	1	1	7	0	1	1	1	7	0	1	1	1	7
1	0	0	0	8	1	0	0	0	8	1	0	0	0	8	1	0	0	0	8
1	0	0	1	9	1	0	0	1	9	1	0	0	1	9	1	0	0	1	9
1	0	1	0	A	1	0	1	0	A	1	0	1	0	A	1	0	1	0	A
1	0	1	1	B	1	0	1	1	B	1	0	1	1	B	1	0	1	1	B
1	1	0	0	C	1	1	0	0	C	1	1	0	0	C	1	1	0	0	C
1	1	0	1	D	1	1	0	1	D	1	1	0	1	D	1	1	0	1	D
1	1	1	0	E	1	1	1	0	E	1	1	1	0	E	1	1	1	0	E
1	1	1	1	F	1	1	1	1	F	1	1	1	1	F	1	1	1	1	F

Respuesta:

1 carácter terminado con un carriage return (**<CR>**) , donde: **0** error , **1** sin error

Ejemplos

Enviar el valor de las 16 salidas de un modulo configurado como modulo **0** , donde:

o15	o14	o13	o12	o11	o10	o9	o8	o7	o6	o5	o4	o3	o2	o1	o0
1	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1

Enviar: **b093D1<CR>**

Respuesta: **1<CR>**

Enviar el valor de las 16 salidas de un modulo configurado como modulo **1** , donde:

o15	o14	o13	o12	o11	o10	o9	o8	o7	o6	o5	o4	o3	o2	o1	o0
0	1	0	1	1	0	0	1	0	1	1	0	1	0	1	0

Enviar: **b1596A<CR>**

Respuesta: **1<CR>**



Especificaciones:

Resistencia del pot: 5 kOhm
 Resolución: 16 bits
 Cantidad máxima de módulos: 8

Comunicación:

Enviar 3 caracteres

El primer carácter representa el tipo de modulo, en este caso seria una **c**
 El segundo carácter representa el numero de modulo, el cual seria un valor entre **0** y **7**
 El tercer carácter representa el terminador, el cual seria un carriage return (**<CR>**)

Respuesta:

12 caracteres hexadecimales , terminados con un carriage return (**<CR>**) , donde :

Los primeros 4 caracteres representan el valor de **c0**
 Los segundos 4 caracteres representan el valor de **c1**
 Los terceros 4 caracteres representan el valor de **c2**

Ejemplos

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo **0**

Enviar: **c0<CR>**
 Respuesta: **04003F56FF5<CR>**

Los primeros 4 caracteres **0400 (1024 en decimal)** representan el valor de **c0**
 Los segundos 4 caracteres **3F56 (16214 en decimal)** representan el valor de **c1**
 Los terceros 4 caracteres **6FF5 (28661 en decimal)** representan el valor de **c2**
 El ultimo carácter representa el terminador (**<CR>**)

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo **1**

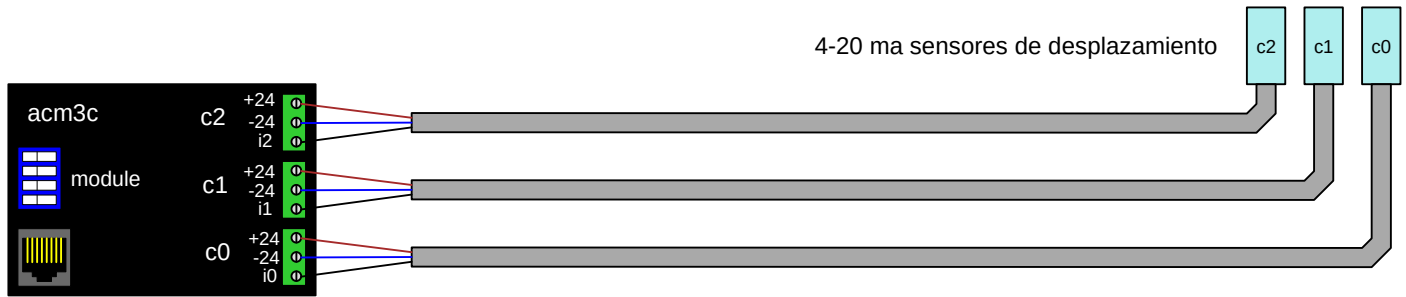
Enviar: **c1<CR>**
 Respuesta: **3398A0F30FFF<CR>**

Los primeros 4 caracteres **3398 (13208 en decimal)** representan el valor de **c0**
 Los segundos 4 caracteres **A0F3 (41203 en decimal)** representan el valor de **c1**
 Los terceros 4 caracteres **0FFF (4095 en decimal)** representan el valor de **c2**
 El ultimo carácter representa el terminador (**<CR>**)

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo **2**

Enviar: **c2<CR>**
 Respuesta: **1516FF00FACC<CR>**

Los primeros 4 caracteres **1516 (5398 en decimal)** representan el valor de **c0**
 Los segundos 4 caracteres **FF00 (65280 en decimal)** representan el valor de **c1**
 Los terceros 4 caracteres **FACC (64204 en decimal)** representan el valor de **c2**
 El ultimo carácter representa el terminador (**<CR>**)



Especificaciones:

Voltaje de salida para alimentar sensor: 24 vdc

Rango de entrada: 4 - 20 mA

Resolución: 16 bits

Cantidad máxima de módulos: 8

Comunicación:

Enviar 3 caracteres

El primer carácter representa el tipo de modulo, en este caso seria una **d**

El segundo carácter representa el numero de modulo, el cual seria un valor entre **0** y **7**

El tercer carácter representa el terminador, el cual seria un carriage return (**<CR>**)

Respuesta:

12 caracteres hexadecimales , terminados con un carriage return (**<CR>**) , donde :

Los primeros 4 caracteres representan el valor de **c0**

Los segundos 4 caracteres representan el valor de **c1**

Los terceros 4 caracteres representan el valor de **c2**

Ejemplos

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo **0**

Enviar: **d0<CR>**

Respuesta: **04003F56FF5<CR>**

Los primeros 4 caracteres **0400 (1024 en decimal)** representan el valor de **c0**

Los segundos 4 caracteres **3F56 (16214 en decimal)** representan el valor de **c1**

Los terceros 4 caracteres **6FF5 (28661 en decimal)** representan el valor de **c2**

El ultimo carácter representa el terminador (**<CR>**)

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo **1**

Enviar: **d1<CR>**

Respuesta: **3398A0F30FFF<CR>**

Los primeros 4 caracteres **3398 (13208 en decimal)** representan el valor de **c0**

Los segundos 4 caracteres **A0F3 (41203 en decimal)** representan el valor de **c1**

Los terceros 4 caracteres **0FFF (4095 en decimal)** representan el valor de **c2**

El ultimo carácter representa el terminador (**<CR>**)

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo **2**

Enviar: **d2<CR>**

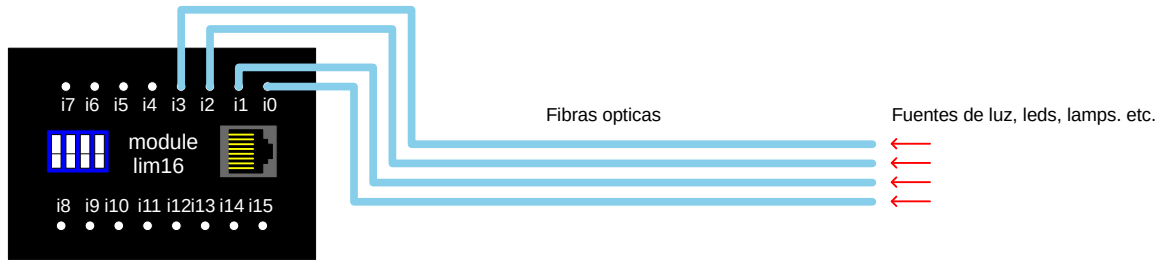
Respuesta: **1516FF00FACC<CR>**

Los primeros 4 caracteres **1516 (5398 en decimal)** representan el valor de **c0**

Los segundos 4 caracteres **FF00 (65280 en decimal)** representan el valor de **c1**

Los terceros 4 caracteres **FACC (64204 en decimal)** representan el valor de **c2**

El ultimo carácter representa el terminador (**<CR>**)



Especificaciones:

Entradas de luz: 16
Cantidad máxima de módulos: 8

Comunicación:

Enviar 3 caracteres:

El primer carácter representa el tipo de modulo, en este caso seria una **e**
El segundo carácter representa el numero de modulo, el cual seria un valor entre **0** y **7**
El tercer carácter representa el terminador, el cual seria un carriage return (**<CR>**)

Respuesta:

4 caracteres hexadecimales terminados con un carriage return (**<CR>**) , donde:

Los primeros 4 caracteres representan el valor de las 16 entradas , (0000 a FFFF)
El último carácter representa el terminador (**<CR>**)

Ejemplos

Leer el valor de las 16 entradas de un modulo configurado como modulo **0**

Enviar: **e0<CR>**
Respuesta: **0000<CR>**

Los primeros 4 caracteres **0000** , en decimal seria un valor de **0**
El último carácter representa el terminador (**<CR>**)

Leer el valor de voltaje de un modulo configurado como modulo **1**

Enviar: **e1<CR>**
Respuesta: **8000<CR>**

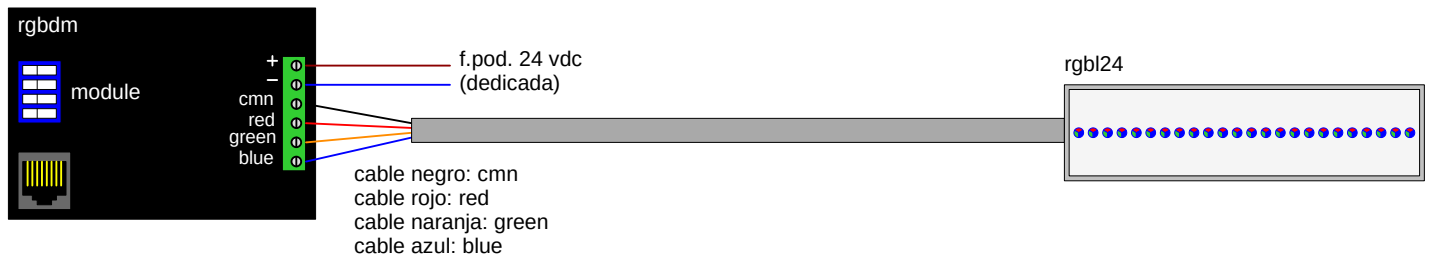
Los primeros 4 caracteres **8000** , en decimal seria un valor de **32768**
El último carácter representa el terminador (**<CR>**)

Para determinar el valor de i0 , si el valor decimal **AND 1 = 1** , entonces i0 = 1 , de lo contrario i0 = 0
Para determinar el valor de i1 , si el valor decimal **AND 2 = 2** , entonces i1 = 1 , de lo contrario i1 = 0
Para determinar el valor de i2 , si el valor decimal **AND 4 = 4** , entonces i2 = 1 , de lo contrario i2 = 0
Para determinar el valor de i3 , si el valor decimal **AND 8 = 8** , entonces i3 = 1 , de lo contrario i3 = 0

Para determinar el valor de i4 , si el valor decimal **AND 16 = 16** , entonces i4 = 1 , de lo contrario i4 = 0
Para determinar el valor de i5 , si el valor decimal **AND 32 = 32** , entonces i5 = 1 , de lo contrario i5 = 0
Para determinar el valor de i6 , si el valor decimal **AND 64 = 64** , entonces i6 = 1 , de lo contrario i6 = 0
Para determinar el valor de i7 , si el valor decimal **AND 128 = 128** , entonces i7 = 1 , de lo contrario i7 = 0

Para determinar el valor de i8 , si el valor decimal **AND 256 = 256** , entonces i8 = 1 , de lo contrario i8 = 0
Para determinar el valor de i9 , si el valor decimal **AND 512 = 512** , entonces i9 = 1 , de lo contrario i9 = 0
Para determinar el valor de i10 , si el valor decimal **AND 1024 = 1024** , entonces i10 = 1 , de lo contrario i10 = 0
Para determinar el valor de i11 , si el valor decimal **AND 2048 = 2048** , entonces i11 = 1 , de lo contrario i11 = 0

Para determinar el valor de i12 , si el valor decimal **AND 4096 = 4096** , entonces i12 = 1 , de lo contrario i12 = 0
Para determinar el valor de i13 , si el valor decimal **AND 8192 = 8192** , entonces i13 = 1 , de lo contrario i13 = 0
Para determinar el valor de i14 , si el valor decimal **AND 16384 = 16384** , entonces i14 = 1 , de lo contrario i14 = 0
Para determinar el valor de i15 , si el valor decimal **AND 32768 = 32768** , entonces i15 = 1 , de lo contrario i15 = 0



Especificaciones:

Colores: Rojo , verde , azul , cian , magenta , amarillo y blanco

Cantidad máxima de módulos: 8

Cantidad máxima de barras conectadas: No rebasar 2.0 amps de consumo (vea consumo de corriente, en pagina siguiente)

Comunicación:

Enviar 4 caracteres:

El primer carácter representa el tipo de modulo, en este caso seria una **f**

El segundo carácter representa el numero de modulo, el cual seria un valor entre **0** y **7**

El tercer carácter representa el valor del color (R=rojo, G=verde, B=azul, C=cian, M=magenta, Y=amarillo, W=blanco y 0=Apagar)

El ultimo carácter representa el terminador, el cual seria un carriage return (**<CR>**)

Respuesta:

1 carácter terminado con un carriage return (**<CR>**) , donde: **0** error , **1** sin error

Ejemplos

Enviar el valor del color rojo , de un modulo configurado como **0**

Enviar: **f0R<CR>**

Respuesta: **1<CR>**

Enviar el valor del color verde , de un modulo configurado como **1**

Enviar: **f1G<CR>**

Respuesta: **1<CR>**

Enviar el valor del color azul , de un modulo configurado como **2**

Enviar: **f2B<CR>**

Respuesta: **1<CR>**

Enviar el valor del color cian , de un modulo configurado como **3**

Enviar: **f3C<CR>**

Respuesta: **1<CR>**

Enviar el valor del color magenta , de un modulo configurado como **4**

Enviar: **f4M<CR>**

Respuesta: **1<CR>**

Enviar el valor del color amarillo , de un modulo configurado como **5**

Enviar: **f5Y<CR>**

Respuesta: **1<CR>**

Enviar el valor del color blanco , de un modulo configurado como **6**

Enviar: **f6W<CR>**

Respuesta: **1<CR>**

Enviar el valor de apagar , de un modulo configurado como **7**

Enviar: **f7O<CR>**

Respuesta: **1<CR>**

Barras de luz multicolor

rgb1b24



Especificaciones:

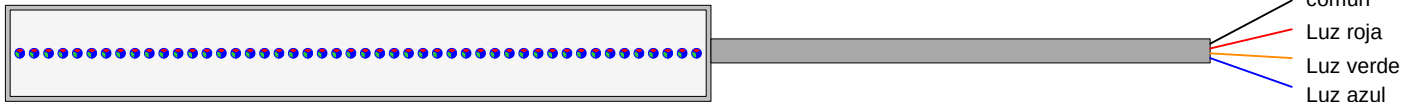
Colores: Rojo, Verde, Azul, Amarillo, Ciano, Magenta y Blanco

Voltaje: 24 vdc

Consumo de corriente: 0.240 amps

Dimensiones: Largo 23cm, ancho 1.7cm, altura 1.2cm

rgb1b48



Especificaciones:

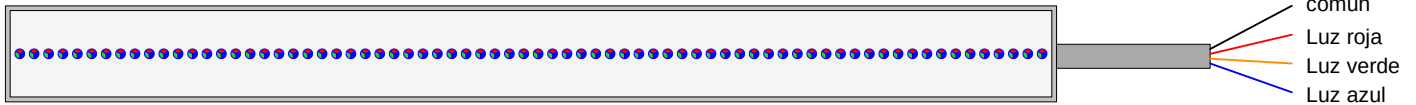
Colores: Rojo, Verde, Azul, Amarillo, Ciano, Magenta y Blanco

Voltaje: 24 vdc

Consumo de corriente: 0.480 amps

Dimensiones: Largo 43 cm, ancho 1.7 cm, altura 1.2 cm

rgb1b72



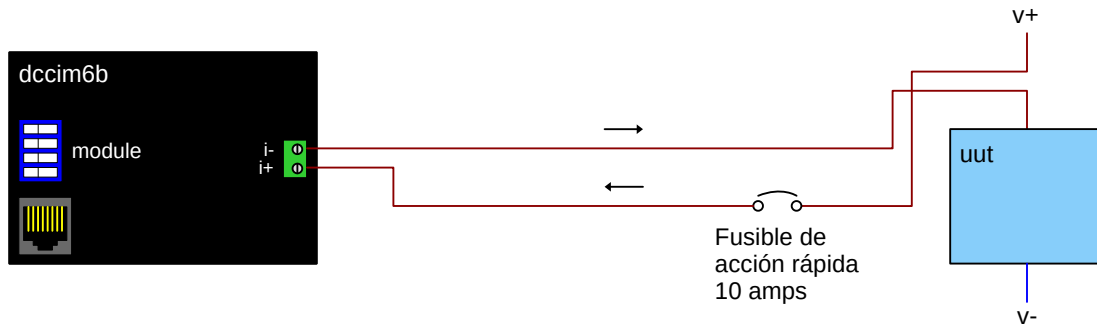
Especificaciones:

Colores: Rojo, Verde, Azul, Amarillo, Ciano, Magenta y Blanco

Voltaje: 24 vdc

Consumo de corriente: 0.720 amps

Dimensiones: Largo 63 cm, ancho 1.7 cm, altura 1.2 cm

**Especificaciones:**

Rango: -6.0 a 0.0 a +6.0 amps bipolar DC
 Resistencia del shunt : 0.0018 Ohms
 Resolución: 16 bits

Cantidad máxima de módulos: 8

Comunicación:**Enviar 3 caracteres**

El primer carácter representa el tipo de modulo, en este caso seria una **g**
 El segundo carácter representa el numero de modulo, el cual seria un valor entre **0** y **7**
 El tercer carácter representa el terminador, el cual seria un carriage return (**<CR>**)

Respuesta:

4 caracteres hexadecimales que representan el resultado de corriente, terminados con un carriage return (**<CR>**)
 Estos 4 caracteres representan un valor de 0 a 65535 en decimal. Para calcular el valor de corriente aplicar lo siguiente:

Si el valor decimal es mayor de 32767
 entonces

$$\text{corriente} = ((\text{valor en decimal} - 32768) / 32767.0) * 6$$

 de lo contrario

$$\text{corriente} = ((\text{valor en decimal} - 32767) / 32767.0) * 6$$

Ejemplos**Leer el valor de corriente de un modulo configurado como modulo 0**

Enviar: **g0<CR>**
 Respuesta: **0000<CR>**

Los primeros 4 caracteres **0000 (0 en decimal)** representan el valor de **corriente**
 El ultimo carácter representa el terminador (**<CR>**)

$$\text{corriente} = ((\text{valor en decimal} - 32767) / 32767.0) * 6 = ((0 - 32767) / 32767.0) * 6 = (-32767/32767) * 6 = -1.0 * 6 = -6.00 \text{ Amps}$$

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo 1

Enviar: **g1<CR>**
 Respuesta: **8000<CR>**

Los primeros 4 caracteres **8000 (32768 en decimal)** representan el valor de **corriente**
 El ultimo carácter representa el terminador (**<CR>**)

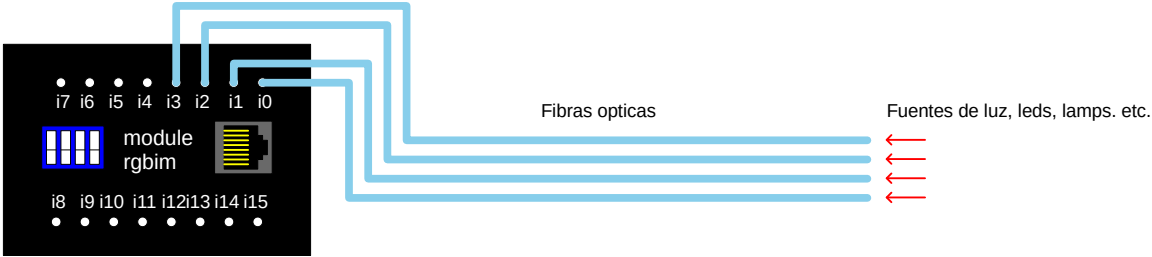
$$\text{corriente} = ((\text{valor decimal} - 32768) / 32767.0) * 6 = (32768 - 32768) / 32767.0 * 6 = (0 / 32767.0) * 6 = 0.00 * 6 = 0.00 \text{ amps}$$

Leer el valor de las coordenadas c0, c1 y c2 de un modulo configurado como modulo 2

Enviar: **g2<CR>**
 Respuesta: **FFFF<CR>**

Los primeros 4 caracteres **FFFF (65535 en decimal)** representan el valor de **corriente**
 El ultimo carácter representa el terminador (**<CR>**)

$$\text{corriente} = ((\text{valor decimal} - 32768) / 32767.0) * 6 = (65535 - 32768) / 32767.0 * 6 = (32767 / 32767.0) * 6 = 1.0 * 6 = +6.00 \text{ amps}$$



Especificaciones:

Entradas de luz por modulo: 16

Cantidad máxima de módulos: 16

Cantidad máxima de entradas: 256

Comunicación:

Enviar 3 caracteres:

El primer carácter representa el tipo de modulo, en este caso seria una **i**

El segundo carácter representa el numero de modulo, el cual seria un valor entre **0** y **F** (carácter hexadecimal)

El tercer carácter representa el terminador, el cual sería un carriage return (**<CR>**)

Respuesta:

96 caracteres hexadecimales terminados con un carriage return (<CR>), donde:

Los primeros 2 caracteres representan el valor de la entrada 0, del rojo en hexadecimal (0 a 255 en decimal).

Los siguientes 2 caracteres representan el valor de la entrada 0, del verde en hexadecimal (0 a 255 en decimal).

Los siguientes 2 caracteres representan el valor de la entrada 0, del azul en hexadecimal (0 a 255 en decimal).

Los siguientes 2 caracteres representan el valor de la entrada 1, del rojo en hexadecimal (0 a 255 en decimal).

Los siguientes 2 caracteres representan el valor de la entrada 1, del verde en hexadecimal (0 a 255 en decimal).

Los siguientes 2 caracteres representan el valor de la entrada 1, del azul en hexadecimal (0 a 255 en decimal).

Así consecutivamente hasta llegar a los caracteres correspondientes a los valores de rojo, verde y azul de la entrada 15.

El último carácter representa el terminador (<CR>)

Ejemplos

Leer el valor de las 16 entradas de un modulo configurado como modulo 0

Enviar: i0<CR>

Respuesta:

```
CC32000000000FF0000000000000000000000000000000003AC4004692264A9A1A00FF0000000000FF0000FF0000FF00<CR>
```

Los primeros 2 caracteres **CC** , en decimal seria un valor de **204**, es decir el valor de rojo para la entrada 0 seria de **204**

Los siguientes 2 caracteres **32**, en decimal seria un valor de **50**, es decir el valor de verde para la entrada 0 seria de **50**

Los siguientes 2 caracteres **00**, en decimal seria un valor de **0**, es decir el valor de azul para la entrada 0 seria de **0**

Los siguientes 2 caracteres **00** , en decimal seria un valor de **0**, es decir el valor de rojo para la entrada 1 seria de **204**

Los siguientes 2 caracteres **00**, en decimal seria un valor de **0**, es decir el valor de verde para la entrada 1 seria de **50**

Los siguientes 2 caracteres **00**, en decimal seria un valor de **0**, es decir el valor de azul para la entrada 1 seria de **0**

Así consecutivamente hasta llegar a los caracteres correspondientes a los valores de rojo, verde y azul de la entrada 15.

El último carácter representa el terminador (<CR>)

Comunicación con los módulos

Para comunicarse con los módulos directamente, conecte el hub8u a un puerto USB de una PC, o el hub8r a un puerto serial de un PLC. Use la información descrita en páginas previas, para averiguar la estructura de lo que envía y recibe, sobre el puerto serial en cuestión.

Los parámetros de comunicación si el módulo esta configurado para 115200 bauds son: 115200 8N1

Los parámetros de comunicación si el módulo esta configurado para 38400 bauds son: 38400 8N1

Si su sistema de control esta basado en PC, descargue de la página iomlogic.com

Ioport.x86.zip si va a desarrollar su aplicación a 32 bits

Ioport.x64.zip si va a desarrollar su aplicación a 64 bits

este zip contiene 2 archivos (ioportlib.dll e ioportnet.dll).

Ioportlib.dll es una librería nativa que gestiona la comunicación entre los módulos y su aplicación, esta libreria es thread-safe, es decir puede ser utilizada en aplicaciones multihilo. Si se utiliza en una aplicación multihilo asegurese que solo uno de los hilos es quien abre y cierra el puerto de comunicación.

Ioportnet.dll es una librería net que encapsula las funciones de ioportlib.dll, que nos permite desarrollar nuestra aplicación en el entorno de net (visual c#, visual basic net, etc), de una manera mas amigable.

Para utilizar ioportnet.dll en su aplicación, deberá referenciar esta librería en su proyecto.

A continuación encontrara información de las funciones de ioportnet.dll

Funciones generales

Open

Esta función se usa para abrir el puerto de comunicación asignado al módulo hub8u

```
C# bool Ioportnet.io.Open(string PortName);  
vb.net Ioportnet.io.Open(PortName as String) As Boolean
```

Parametros:

PortName: puerto de comunicación (COM1 a COM99)

Valor de retorno:

True si el puerto ha sido abierto sin problemas, de lo contrario el valor de retorno sera False.

Close

Esta función se usa para cerrar el puerto de comunicación asignado al módulo hub8u

```
C# void Ioportnet.io.Close();  
vb.net Ioportnet.io.Close()
```

Err

Esta función se usa para obtener el ultimo error generado

```
C# string Ioportnet.io.Err();  
vb.net Ioportnet.io.Err() As String
```

Valor de retorno:

Descripción del ultimo error generado

Update

Esta función se usa para actualizar el estado de las entradas digitales

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "dim16"

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar.
Si el valor de retorno fue **False** use la función **GetErr** para obtener la descripción del error.

dim16Val

Esta función se usa para obtener el valor de las entradas digitales

```
C# bool Ioportnet.io.dim16Val(int ModuleNum,int InpNum);
vb.net ioportnet.io.dim16Val(ModuleNum As Integer,InpNum as Integer) As Boolean
```

Parametros:

ModuleNum: Numero de módulo (0 a 7)

InpNum: Numero de entrada (0 a 15)

Valor de retorno:

True si en la entrada hay 24 vdc, y **False** si en la entrada hay 0 vdc

dom16Val

```
C# void Ioportnet.io.dom16Val(int ModuleNum,int OutNum,bool OutVal);
vb.net ioportnet.io.dom16Val(ModuleNum As Integer,OutNum as Integer,OutVal As Boolean)
```

Parametros:

ModuleNum: Numero de módulo (0 a 7)

OutNum: Numero de salida (0 a 15)

OutVal: False para apagar salida, True para encender salida

Update

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "dom16"

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar.
Si el valor de retorno fue **False** use la función **GetErr** para obtener la descripción del error.

Update

Esta función se usa para actualizar el valor de las 3 coordenadas

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "acm3p"

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar. Si el valor de retorno fue **False** use la función **GetErr** para obtener la descripción del error.

acm3pVal

Esta función se usa para obtener el valor de las coordenadas

```
C# double Ioportnet.io.acm3pVal(int ModuleNum,int InpNum);
vb.net ioportnet.io.acm3pVal(ModuleNum As Integer,InpNum as Integer) As Double
```

Parametros:

ModuleNum: Numero de módulo (0 a 7)

InpNum: Numero de entrada (0 a 2)

Valor de retorno:

Valor de la coordenada especificada en InpNum, en unidades de porcentaje de posición (0.00 a 100.00)

Update

Esta función se usa para actualizar el valor de las 3 coordenadas

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "acm3c"

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar. Si el valor de retorno fue **False** use la función **GetErr** para obtener la descripción del error.

acm3cVal

Esta función se usa para obtener el valor de las coordenadas

```
C# double Ioportnet.io.acm3cVal(int ModuleNum,int InpNum);
vb.net ioportnet.io.acm3cVal(ModuleNum As Integer,InpNum as Integer) As Double
```

Parametros:

ModuleNum: Numero de módulo (0 a 7)

InpNum: Numero de entrada (0 a 2)

Valor de retorno:

Valor de la coordenada especificada en InpNum, en unidades de porcentaje de posición (0.00 a 100.00)

Update

Esta función se usa para actualizar el estado de las entradas digitales de luz

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "lim16"

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar. Si el valor de retorno fue **False** use la función **GetErr** para obtener la descripción del error.

lim16Val

Esta función se usa para obtener el valor de las entradas digitales de luz

```
C# bool Ioportnet.io.lim16Val(int ModuleNum,int InpNum);
vb.net ioportnet.io.lim16Val(ModuleNum As Integer,InpNum as Integer) As Boolean
```

Parametros:

ModuleNum: Numero de módulo (0 a 7)

InpNum: Numero de entrada (0 a 15)

Valor de retorno:

True si en la entrada hay luz, y **False** si en la entrada no hay luz

rgbdmVal

```
C# void Ioportnet.io.rgbdmVal(int ModuleNum,char Color);
vb.net ioportnet.io.rgbdmVal(ModuleNum As Integer,Color as Char)
```

Parametros:

ModuleNum: Numero de módulo (0 a 7)

Color: Color (R=rojo, G=verde, B=azul, Y=amarillo,C=cian, M=magenta, W=blanco, 0=apagado)

Update

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "rgbdm"

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar. Si el valor de retorno fue **False** use la función **GetErr** para obtener la descripción del error.

Update

Esta función se usa para actualizar el valor de corriente a través del módulo dccim6b

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "dccim6b"

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar.
Si el valor de retorno fue **False** use la función **Err** para obtener la descripción del error.

dccimVal

Esta función se usa para obtener el valor de corriente a través del módulo dccim

```
C# double Ioportnet.io.dccimVal(int ModuleNum);
vb.net ioportnet.io.dccimVal(ModuleNum As Integer) As Double
```

Parametros:

ModuleNum: Numero de módulo (0 a 7)

Valor de retorno:

Valor de corriente medida, en unidades de amperes (-6.0 a 0.0 a +6.0)

Update

Esta función se usa para actualizar los valores de Rojo, Verde y Azul de las 16 entradas del modulo rgbim

```
C# bool Ioportnet.io.Update(string ModuleNam,int ModuleNum);
vb.net ioportnet.io.Update(ModuleNam As String,ModuleNum As Integer) As Boolean
```

Parametros:

ModuleNam: Aqui debera poner "rgbim"

ModuleNum: Numero de módulo (0 a F)

Valor de retorno:

True si se actualizo con éxito, el estado de las entradas, **False** si no se pudieron actualizar.
Si el valor de retorno fue **False** use la función **Err** para obtener la descripción del error.

rgbimVal

Esta función se usa para obtener los valores de Rojo, Verde y Azul de una entrada en particular

```
C# int Ioportnet.io.rgbimVal(int ModuleNum,int InpNum,int ColorNum);
vb.net ioportnet.io.rgbimVal(ModuleNum As Integer,InpNum As Integer,ColorNum As Integer) As Integer
```

Parametros:

ModuleNum: Numero de módulo (0 a 15)

InpNum: Numero de entrada (0 a 15)

ColorNum: Numero de color (0 a 2) [0 para el Rojo, 1 para el Verde y 2 para el Azul]

Valor de retorno:

Valor de color medido (0 a 255)