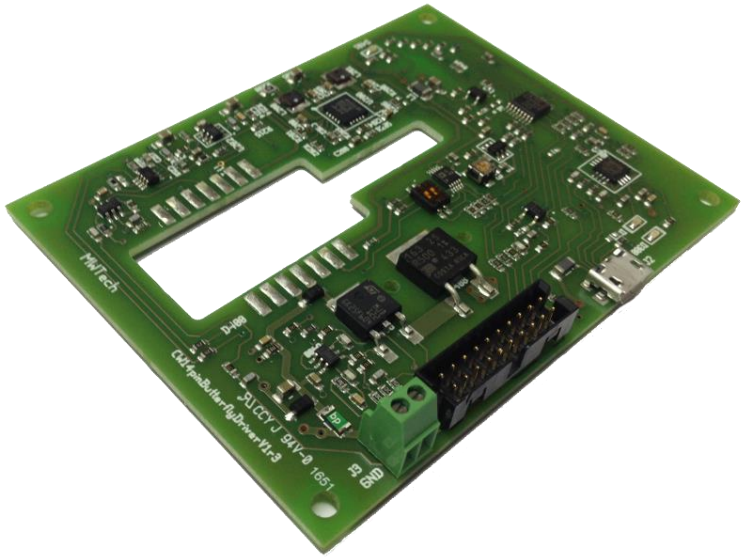
CW14pinButterflyDriver1500

Control Application Guide

Version 1.0.8

Date: 30-June-2017



(This picture is for illustrative purposes only and details may vary)

Innolume GmbH
Konrad-Adenauer-Allee 11
44263 Dortmund
GERMANY

Phone: +49 (0) 231 47730-200
Email: info@innolume.com

Web: www.innolume.com

TABLE OF CONTENTS

IMPORTANT NOTES - PLEASE READ

1. This guide provides general information about Innolume´s CW14pinButterflyDriver1500 Control Application, and the remote commands of the CW14pinButterflyDriver1500.
2. For a detailed description of the CW14pinButterflyDriver1500, refer to the User's Guide, Hardware.

1. Overview

This guide provides general information about Innolume´s CW14pinButterflyDriver1500 Control Application, particularly the computer system requirements for the Control Application, the installation instructions and a general overview of the CW14pinButterflyDriver1500 Control Application. In addition, this guide contains the remote commands for control of the CW driver.

2. System Requirements

The following computer system requirements are needed to install the CW14pinButterflyDriver1500 Control Application:

- Microsoft Windows™ operating system (version XP, Vista, 7, 8, 8.1 or 10);
- USB port or COM port;
- Graphic card capable of a minimal resolution of 800x600 and 256 colors;
- Minimum of 1 GB of system RAM (recommended);
- 650 kB free on disk (plus additional space for Microsoft .NET Framework and for Microsoft Visual C++ Redistributable).

3. Installation Instructions

The installation of the CW14pinButterflyDriver1500 Control Application and USB drivers is simple and straightforward. Before connecting the CW driver to the computer, follow these steps:

1) Close all running applications before proceeding. With the device unplugged, run the installation package (Setup.exe).
2) Follow the instructions on the screen. The CW14pinButterflyDriver1500 Control Application requires the Microsoft .NET Framework and the Microsoft Visual C++ Redistributable. The setup will install compatible versions of the Microsoft .NET Framework and of the Microsoft Visual C++ Redistributable if not already installed on the computer.
3) Once the CW14pinButterflyDriver1500 Control Application is installed, you will be prompted for configuring your connection with the CW14pinButterflyDriver1500 device.
4) Connect the CW14pinButterflyDriver1500 device to the computer.
5) If the device is not automatically detected, restart the computer.

4. Graphical User Interface (GUI)

Figure 1 shows the CW14pinButterflyDriver1500 Control Application GUI. It displays the CW14pinButterflyDriver1500 control functions and the user can, through this GUI:

1) Enable or disable the laser current (software enable);
2) Change and store the setpoint temperature for the TEC control;
3) Set the laser current limit;
4) Change and store the laser current setpoint;
5) Read the state of the overtemperature alarm, the system status and the power indicator levels.

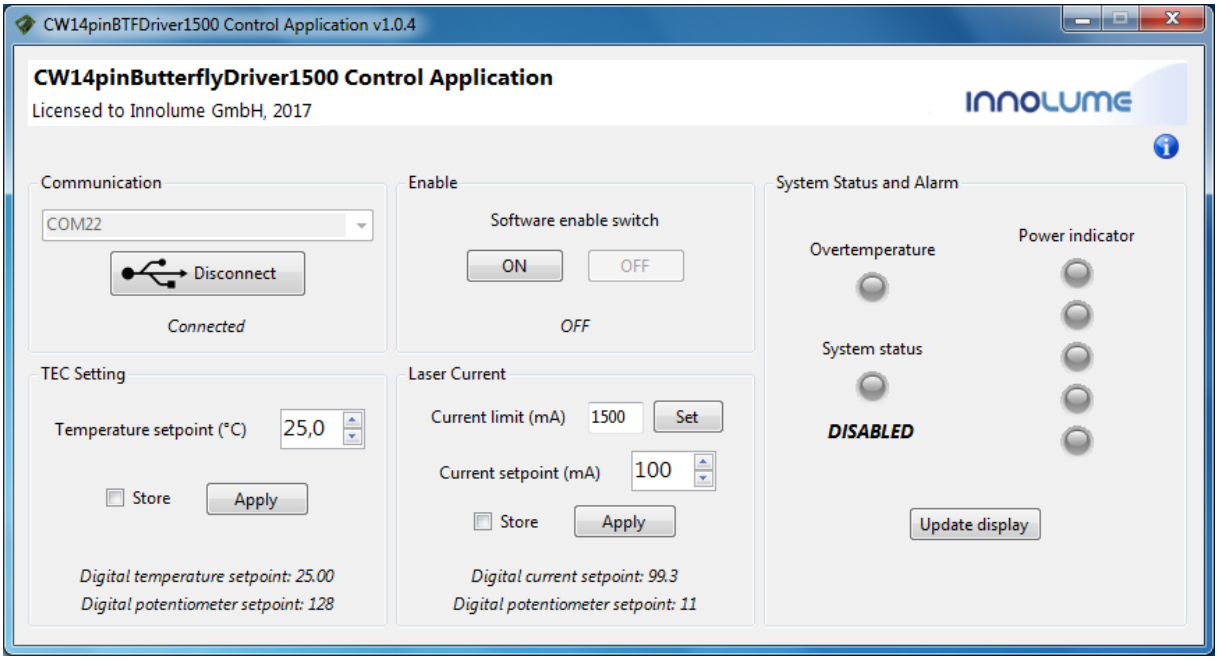


Figure 1 CW14pinButterflyDriver1500 Control Application GUI

Figure 2 shows the "About" dialog window, displaying information about the software Control Application and the company contacts. The user can access this window using the command, located in the top right corner of the GUI window.

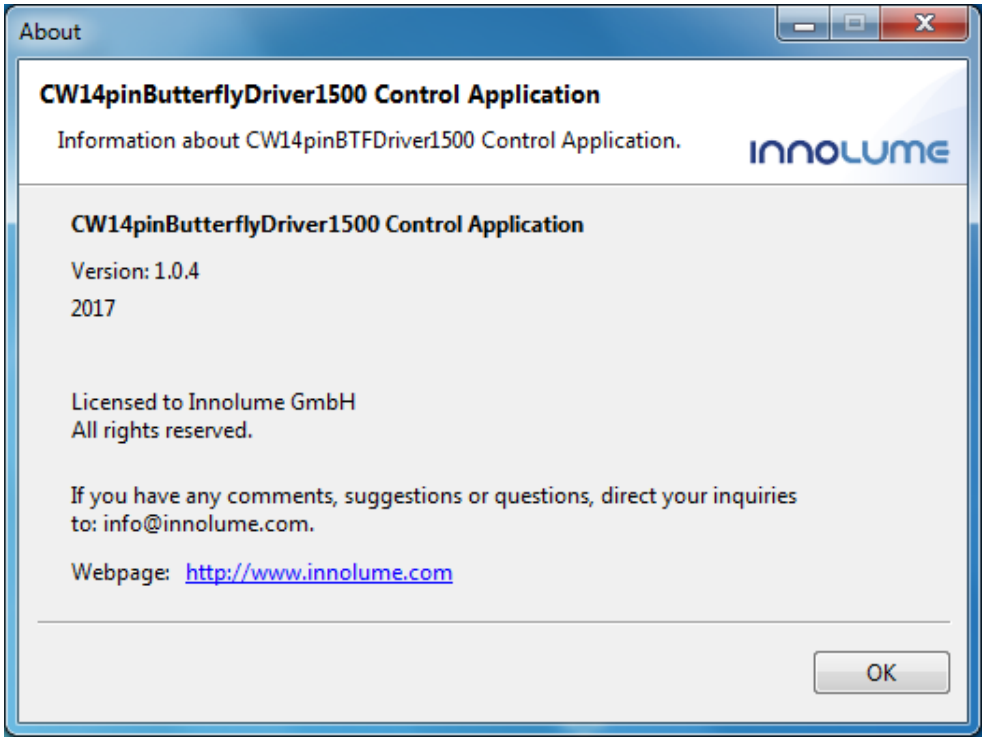


Figure 2 CW14pinButterflyDriver1500 Control Application GUI, About Window

4.1 Communication

The CW14pinButterflyDriver1500 Control Application starts as shown in Figure 3.

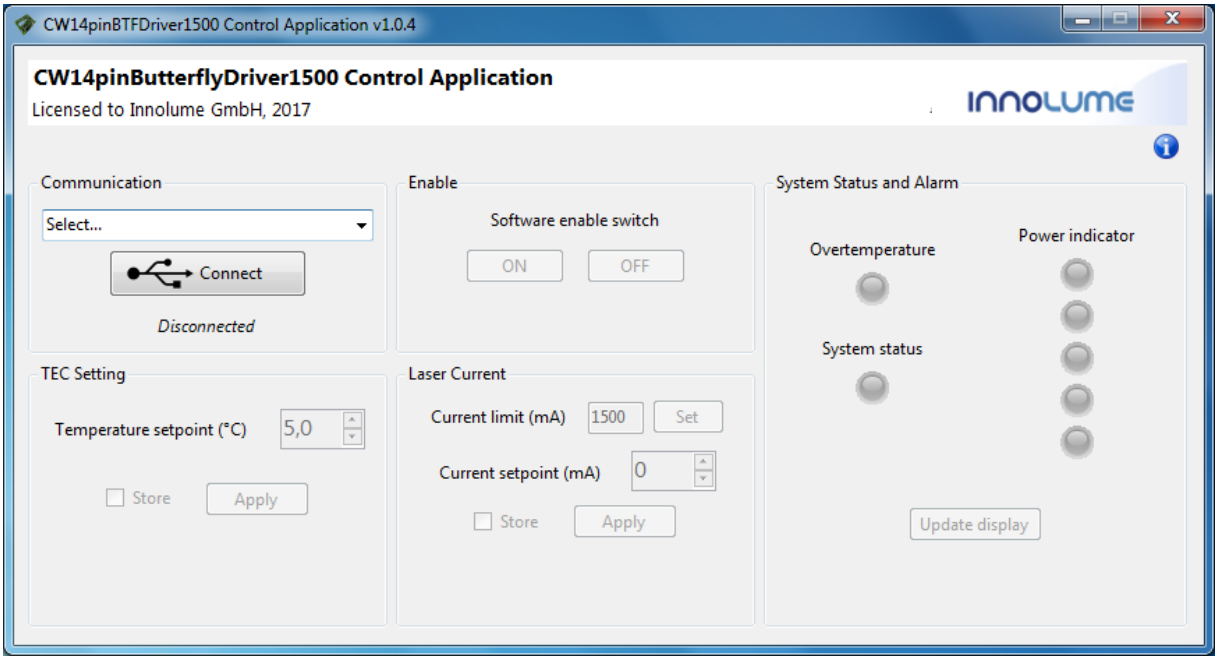


Figure 3 CW14pinButterflyDriver1500 Control Application GUI, startup

The communication between the CW driver and the application can be performed by USB or RS-232. The CW14pinButterflyDriver1500 Control Application finds and lists the COM ports available and, when connected to the computer, the CW driver is detected as COM port and the communication between the software and the device is possible. Figure 4 illustrates the recognition of the COM ports available by the application.

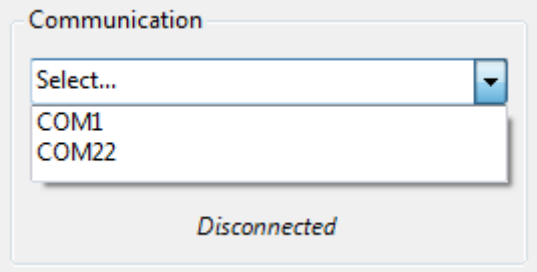


Figure 4 CW14pinButterflyDriver1500 Control Application GUI, connection to CW driver

While the CW14pinButterflyDriver1500 is connecting, the application reads the current values applied to the device and updates the corresponding fields.

4.2 Software Enable

The user can enable/disable the CW driver laser current by software, using the software enable switch contained in the application (Figure 5). The label below presents the current state of the software enable.

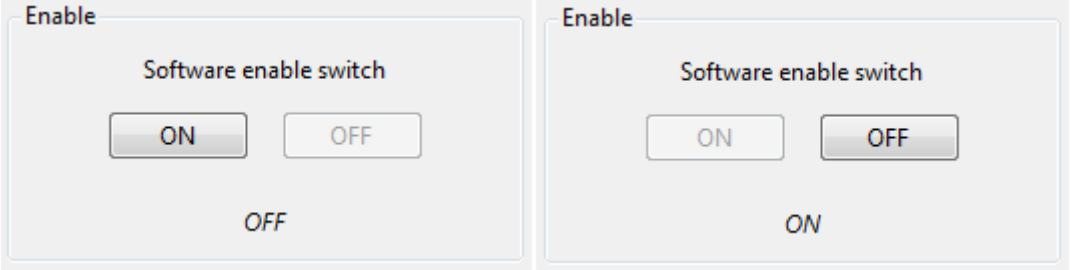


Figure 5 CW14pinButterflyDriver1500 Control Application GUI, software enable

The default state of the software enable switch is the actual state of enable (EN_ext).

4.3 Settings

The settings of CW14pinButterflyDriver1500 Control Application are shown in Figure 6, and Table 1 presents the allowed ranges for different settings.

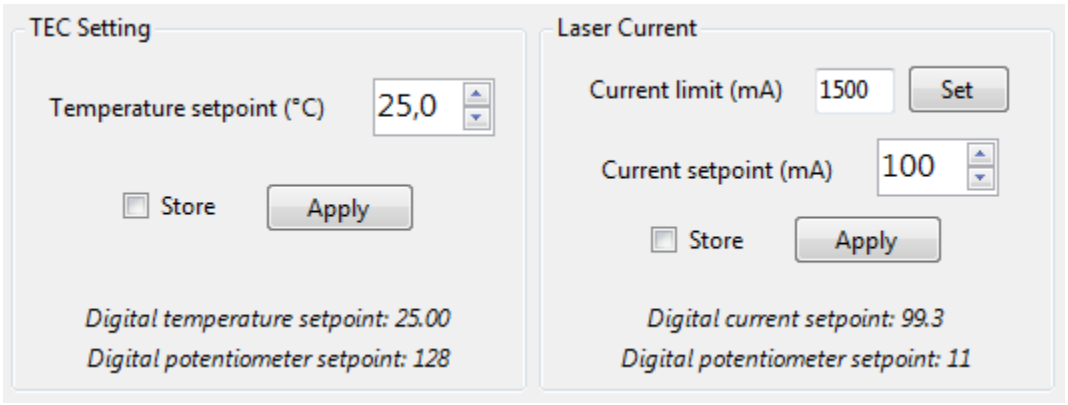


Figure 6 CW14pinButterflyDriver1500 Control Application GUI, settings

Setting	*Range*	
	Minimum	*Maximum*
Temperature setpoint	5 °C	55 °C
Laser current limit	0 mA	1500 mA
Laser current setpoint	0 mA	Laser current limit

Table 1 Ranges for settings of CW14pinButterflyDriver1500 Control Application

The application of the settings is independent of the laser enable state.
The "Store" check boxes allow storing the temperature and the laser current setpoints, and after the power is cycled, the temperature and the laser current keep the last setpoint stored respectively. In the initial power-up, the digital potentiometers setpoints are at midscale (128), for temperature and for laser current. The "Apply" buttons writes and stores (if the box is checked) the current setpoint value to the CW driver. The application displays also the digital potentiometer setpoint for temperature and for laser current. The digital potentiometer is a 256-position digitally-controlled variable resistor, and the digital temperature and laser current setpoints displayed are the digitized values calculated from the correspondent digital potentiometer setpoint.

4.4 System Status and Alarm

The CW14pinButterflyDriver1500 Control Application allows the user to read the states of the overtemperature alarm, of the system status and of the power indicator levels, as shown in Figure 7.

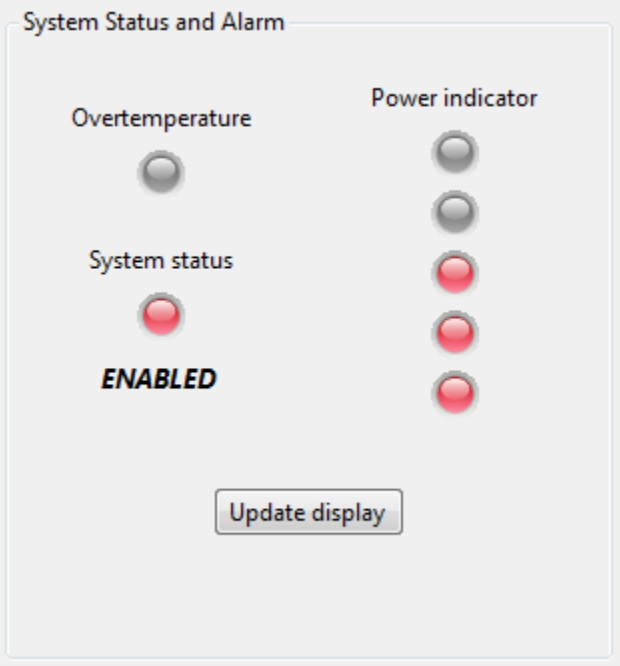


Figure 7 CW14pinButterflyDriver1500 Control Application GUI, system status and alarm

This information is updated when the user clicks on the "Update display" button, on any "Apply" button, or on the "ON" or "OFF" buttons of the software enable switch.

The meaning of the icons used for the overtemperature alarm is:

- Alarm is OFF
- Alarm is ON

For system status, the meaning of the icons is:

- System is disabled
- System is enabled

Relatively to the power indicator levels, the lower level is the lower power (level 1), and the higher level is the higher power (level 5). The meaning of the icons used is:

- Level is OFF
- Level is ON

5. Remote Commands

The user can communicate with the CW driver by USB or RS-232 interfaces. The serial data format is fixed: one start bit, eight data bits, no parity bit, and one stop bit. The default baud rate is 9600 bit/s, it can be changed if needed[1].
The remote commands of CW14pinButterflyDriver1500 are based in ASCII characters. After a power-up, the CW driver will send two continuous bytes to the user to indicate a start-up condition. These two bytes are, in hexadecimal representation, 0x4F and 0x4B ('O' and 'K' in ASCII, respectively).
The CW14pinButterflyDriver1500 has a time-out feature for unfinished commands sequence. The delay between any two bytes of data coming from the user should be less than 655 ms. If this condition is not met, the CW driver will time-out and clear the receive buffer, and then it starts to wait for the next command from the user.
Unrecognized commands are ignored by the CW14pinButterflyDriver1500.

5.1 Laser Current

The laser current setpoint is controlled through a digital potentiometer, which is a 256-position (8-bit) digitally-controlled variable resistor. The relation between the laser current and the digital potentiometer are presented in the functions below, with its parameters described in Table 2.

$$I(bitset) = \frac{kappa \times r \times R_T}{\frac{255}{bitset} + \frac{r \times R_T}{K}} + x_0$$

$$bitset(I) = \frac{255}{\frac{kappa \times r \times R_T}{I - x_0} - \frac{r \times R_T}{K}}$$

Parameter	*Description*	*Range / Value*
I	Laser current setpoint	x_0 – 1500 (mA)
bitset	Digital potentiometer setpoint	0 – 255
R_T	Total resistance of digital potentiometer	10 x (tolerance / 100 + 1) (kΩ)
kappa	Calibration constant	104.17
K	Calibration constant	106.493
r	Calibration constant	1.63172
x_0	Calibration constant	26.7004

Table 2 Parameters description for laser current calculation

The value of total resistance of digital potentiometer, R_T, used in previous functions, is the nominal value (10 kΩ) corrected for tolerance read from the CW driver, to allow more precision in calculations of laser current and of digital potentiometer setpoints.

5.1.1 Read Laser Current Tolerance

The read laser current tolerance command allows reading the tolerance of nominal resistance of digital potentiometer of laser current. Its format is the following:

[1] SC18IM700IPW: Master I²C-bus controller with UART interface.

- CW14pinButterflyDriver1500 receives (where 'S' is 0x53 and 'P' is 0x50, in hexadecimal representation):

'S' | 0x9C | 0x01 | 0x3E | 'S' | 0x9D | 0x02 | 'P'

- CW14pinButterflyDriver1500 transmits:

Data 0 | Data 1

Data 0								***Data 1***							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
Sign	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
Sign	Integer number							Decimal number							

Table 3 Read tolerance for laser current

As shown in Table 3, the MSB of data byte 0 contains the sign (0 = + and 1= −) and the seven LSBs designate the integer portion of the tolerance. In the data byte 1, all eight data bits indicate the decimal portion of tolerance. Note the decimal portion has a limited accuracy of 0.1%.

Example:

Data 0 = 1000 0001

MSB: 1 = -

7 LSB: 000 0001 = 1

Data 1 = 1101 1100

8 LSB: 1101 1100 = 220 x 2^{-8} = 0.86

Tolerance = -1.86%

Rounded tolerance = -1.9%

R_T = 9.810 kΩ

5.1.2 Read Laser Current Setpoint

The read laser current setpoint command allows reading the digital potentiometer setpoint of laser current. Its format is the following:

- CW14pinButterflyDriver1500 receives:

'S' | 0x9C | 0x01 | 0x00 | 'S' | 0x9D | 0x01 | 'P'

- CW14pinButterflyDriver1500 transmits:

Data

Example:

Data = 1000 0000 = Digital potentiometer setpoint = 128

Laser current setpoint = 819.0 mA (with R_T = 10 kΩ)

5.1.3 Read Laser Current Setpoint Stored

The read laser current setpoint stored command allows reading the digital potentiometer setpoint of laser current stored. Its format is the following:

- CW14pinButterflyDriver1500 receives:

‘S’ | 0x9C | 0x01 | 0x20 | ‘S’ | 0x9D | 0x01 | ‘P’

- CW14pinButterflyDriver1500 transmits:

Data

Example:
Data = 0100 1010 = Digital potentiometer setpoint stored = 74
Laser current setpoint stored = 499.0 mA (with R_T = 10 kΩ)

5.1.4 Write Laser Current Setpoint

The write laser current setpoint command allows writing the digital potentiometer setpoint of laser current. Its format is the following:

- CW14pinButterflyDriver1500 receives:

‘S’ | 0x9C | 0x02 | 0x00 | Data | ‘P’

Example:
Laser current setpoint to write = 1000 mA
Digital potentiometer setpoint = 160 = Data = 1010 0000 (with R_T = 10 kΩ)

5.1.5 Store Laser Current Setpoint

The store laser current setpoint command allows storing the digital potentiometer setpoint of laser current applied currently. Its format is the following:

- CW14pinButterflyDriver1500 receives:

‘S’ | 0x9C | 0x01 | 0xC0 | ‘P’

5.2 TEC

As the laser current, the temperature setpoint is also defined with the use of a 256-position (8-bit) digital potentiometer. The following equations present the relation between the temperature and the digital potentiometer, and its parameters are described in Table 4.

$$T(RNTCround) = \frac{1}{\frac{1}{\beta} \times \log\left(\frac{RNTCround}{RT_0}\right) + \frac{1}{T_0}} - K_0$$

$$RNTCround(bitset) = R210 \times \frac{R218 + R_T \times \frac{birange - bitset}{bitrange}}{R_T + R219 - R_T \times \frac{birange - bitset}{bitrange}}$$

$$bitset(RWA) = bitrange - \frac{bitrange \times RWA}{R_T}$$

$$RWA(RNTC) = \frac{R_T + R218 + R219}{\frac{R210}{RNTC} + 1} - R218$$

$$RNTC(T) = RT_0 \times e^{\beta \times \left(\frac{1}{T + K_0} - \frac{1}{T_0}\right)}$$

Parameter	*Description*	*Range / Value*
T	Temperature setpoint	-7.6 – 66.6 (°C)
bitset	Digital potentiometer setpoint	0 – 255
R_T	Total resistance of digital potentiometer	10 x (tolerance / 100 + 1) (kΩ)
β	Calibration constant	3375
bitrange	Calibration constant	256
K_0	Calibration constant	273.16
R210	Calibration constant	10
R218	Calibration constant	3.3
R219	Calibration constant	3.3
RT_0	Calibration constant	10
T_0	Calibration constant	25 + K_0

Table 4 Parameters description for temperature calculation

For more precision in calculations of temperature and of digital potentiometer setpoints, the nominal resistance of digital potentiometer value (10 kΩ) should be corrected for tolerance read from the CW driver.

5.2.1 Read Temperature Tolerance

The read temperature tolerance command allows reading the tolerance of nominal resistance of digital potentiometer of temperature. Its format is the following:

- CW14pinButterflyDriver1500 receives:

‘S’ | 0x30 | 0x01 | 0x3E | ‘S’ | 0x31 | 0x02 | ‘P’

- CW14pinButterflyDriver1500 transmits:

Data 0 | Data 1

Data 0								*Data 1*							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
Sign	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
Sign	Integer number							Decimal number							

Table 5 Read tolerance for temperature

Table 5 shows the bit description for the data bytes of the tolerance. The data byte 0 has the sign (0 = + and 1 = –) (MSB) and the integer portion (7 LSB), and the data byte 1 designates the decimal portion of tolerance, with a limited accuracy of 0.1%.

Example:

Data 0 = 1000 0001

MSB: 1 = -
7 LSB: 000 0001 = 1
Data 1 = 1101 1100
8 LSB: 1101 1100 = 220 x 2^{-8} = 0.86
Tolerance = -1.86%
Rounded tolerance = -1.9%
R_T = 9.810 kΩ

5.2.2 Read Temperature Setpoint

The read temperature setpoint command allows reading the digital potentiometer setpoint of temperature. Its format is the following:

- CW14pinButterflyDriver1500 receives:

'S' | 0x30 | 0x01 | 0x00 | 'S' | 0x31 | 0x01 | 'P'

- CW14pinButterflyDriver1500 transmits:

Data

Example:
Data = 1000 0000 = Digital potentiometer setpoint = 128
Temperature setpoint = 25.00 °C (with R_T = 10 kΩ)

5.2.3 Read Temperature Setpoint Stored

The read temperature setpoint stored command allows reading the digital potentiometer setpoint of temperature stored. Its format is the following:

- CW14pinButterflyDriver1500 receives:

'S' | 0x30 | 0x01 | 0x20 | 'S' | 0x31 | 0x01 | 'P'

- CW14pinButterflyDriver1500 transmits:

Data

Example:
Data = 0110 1100 = Digital potentiometer setpoint stored = 108
Temperature setpoint stored = 20.11 °C (with R_T = 10 kΩ)

5.2.4 Write Temperature Setpoint

The write temperature setpoint command allows writing the digital potentiometer setpoint of temperature. Its format is the following:

- CW14pinButterflyDriver1500 receives:

'S' | 0x30 | 0x02 | 0x00 | Data | 'P'

Example:

Temperature setpoint to write = 30.0 °C
Digital potentiometer setpoint = 148 = Data = 1001 0100 (with R_T = 10 kΩ)

5.2.5 Store Temperature Setpoint

The store temperature setpoint command allows storing the digital potentiometer setpoint of temperature applied currently. Its format is the following:

- CW14pinButterflyDriver1500 receives:

'S' | 0x30 | 0x01 | 0xC0 | 'P'

5.3 Enable, Overtemperature Alarm and Power Indicator Levels

Figure 8 presents a simplified schematic about the enable configuration.

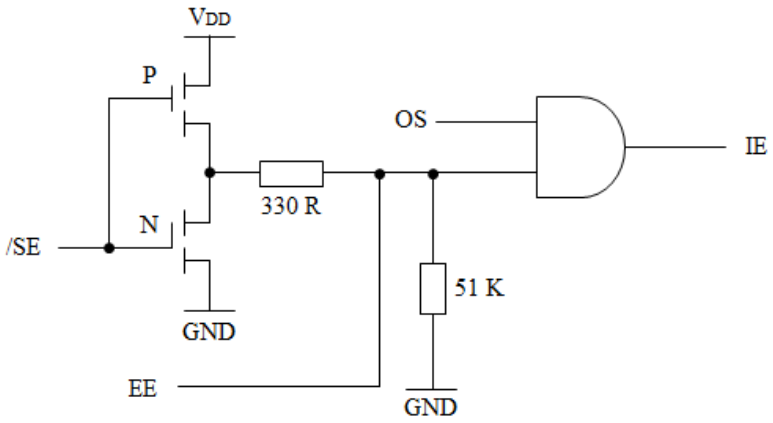


Figure 8 Enable configuration

The CW14pinButterflyDriver1500 is only enabled when the overtemperature shutdown (OS) alarm is high (alarm is not active) and the software and external enable are not pulling down. The CW driver has a register called IOState Register, which bit description is shown in Table 6.

IOState Register							
D7	D6	D5	D4	D3	D2	D1	D0
PL5	SE / EB	PL3	PL4	IE	OS	PL2	PL1

Table 6 IOState Register bit desciption

Using this register, the user can:

- Read:
  - The internal enable (IE), it is the state of the device;
  - The enable bus (EB);
  - The overtemperature alarm;
  - The power indicator levels (PL) (it needs to be configured).
- Write:
  - The software enable (SE) (it needs to be configured);

- The power indicator levels (it needs to be configured).

By default, the data bits of IOState Register are not configured to allow the writing of the software enable and of the power indicator levels. So, to be able to these functions, the user must configure the IOState Register using the configure IOState Register command.

5.3.1 Configure IOState Register

The configure IOState Register command allows configuring the IOState Register to allow the writing of the software enable and of the power indicator levels. Its format is the following:

- CW14pinButterflyDriver1500 receives (where 'W' is 0x57, in hexadecimal representation):

'W' | 0x02 | 0x5F | 0x03 | 0xEF | 'P'

5.3.2 Write IOState Register

The write IOState Register command allows writing the software enable and the power indicator levels of the IOState Register. Its format is the following:

- CW14pinButterflyDriver1500 receives (where 'O' is 0x4F, in hexadecimal representation):

'O' | Data | 'P'

Data							
D7	D6	D5	D4	D3	D2	D1	D0
PL5	SE	PL3	PL4	X	X	PL2	PL1

Table 7 Write IOState Register

As shown in Table 7, using this command does not change the states of the internal enable (D3) and of the overtemperature alarm (D2) (they are don't cares, "X").

For software enable, writing:
'0' - Software enable OFF
'1' - Software enable ON

Relatively to the power indicator levels, writing:
'0' - Level OFF
'1' - Level ON

Note that for this command has effect, the user must configure first the IOState Register using the configure IOState Register command described previously.

5.3.3 Read IOState Register

The read IOState Register command allows reading the internal enable, the enable bus, the overtemperature alarm and the power indicator levels of the IOState Register. Its format is the following:

- CW14pinButterflyDriver1500 receives (where 'I' is 0x49, in hexadecimal representation):

'I' | 'P'

- CW14pinButterflyDriver1500 transmits:

Data

Data							
D7	D6	D5	D4	D3	D2	D1	D0
PL5	EB	PL3	PL4	IE	OS	PL2	PL1

Table 8 Read IOState Register

Reading the internal enable as:
'0' - System is disabled
'1' - System is enabled

For enable bus, reading:
'0' - Enable bus is OFF
'1' - Enable bus is ON

Relatively to the overtemperature alarm, reading as:
'0' - Alarm is ON
'1' - Alarm is OFF

The power indicator of CW14pinButterflyDriver1500 has five levels (PL1 – PL5), which the power level 1 is the lower power, and the power level 5 is the higher power. Reading the power indicator level as:
'0' - Level is OFF
'1' - Level is ON

Once the power indicator levels are Schmitt triggered inputs that also have a glitch suppression circuit, the user should follow the next steps to read correctly its states:
1) Set low level for all power indicator levels (using the write IOState Register command);
2) Set high level for all power indicator levels (using the write IOState Register command);
3) Read the power indicator levels (using the read IOState Register command).

5.4 Python Example

The following Python script is a simple example how to control the CW14pinButterflyDriver1500. It has three parameters (constants) to set:
- Laser current setpoint (`I`);
- Temperature setpoint (`T`);
- Software enable (`OnOff`).

The code executes the next functions:
1) Enable / Disable the laser current, dependent of the `OnOff` parameter;
2) Set the laser current setpoint defined (`I`);

3) Set the temperature setpoint defined (T);
4) Read the IOState Register.

The Python example is based in version Python 2.7.11, that is available for downloading in https://www.python.org/downloads/release/python-2711/. For serial communication, it uses the Python Serial Port Extension version pyserial 2.7, available for downloading in https://pypi.python.org/pypi/pyserial/2.7.

Python example:

```
import serial
import time
import math

# Parameters to set
I = 100 # Laser current setpoint, in mA
T = 25.0 # Temperature setpoint, in C
OnOff = 0x00 # Software enable; For OFF, OnOff = 0x00, for ON, OnOff = 0x40

# Laser current
RT_I = 10.0 # Nominal resistance of digital potentiometer, in kOhm. For
improve precision, correct for tolerance (see CW14pinButterflyDriver1500
Control Application Guide)
kappa = 104.17 # Calibration constant
K = 106.493 # Calibration constant
r = 1.63172 # Calibration constant
x0 = 26.7004 # Calibration constant
bitset_I = int(round(255 / ((kappa * r * RT_I) / (I - x0) - (r * RT_I) /
K), 0)) # Digital potentiometer setpoint
print("I = " + str(I))
print("bitset_I = " + str(bitset_I))

# TEC
RT_T = 10.0 # Nominal resistance of digital potentiometer, in kOhm. For
improve precision, correct for tolerance (see CW14pinButterflyDriver1500
Control Application Guide)
Beta = 3375.0 # Calibration constant
bitrange = 256 # Calibration constant
K0 = 273.16 # Calibration constant
R210 = 10.0 # Calibration constant
R218 = 3.3 # Calibration constant
R219 = 3.3 # Calibration constant
RT0 = 10.0 # Calibration constant
T0 = 25.0 + K0 # Calibration constant
RNTC = RT0 * math.exp(Beta * (1 / (T + K0) - 1 / T0))
RWA = (RT_T + R218 + R219) / (R210 / RNTC + 1) - R218
bitset_T = int(round(bitrange - (bitrange * RWA) / RT_T, 0)) # Digital
potentiometer setpoint
print("T = " + str(T))
print("bitset_T = " + str(bitset_T))

# Serial port
ser = serial.Serial('COM22') # Open serial port which is connected to the
device
print("Serial port used: " + ser.name) # Check which port was really used

# Configure IOState Register to allow the writing of the software enable
and of the power indicator levels
```

```
ser.write((chr(0x57) + chr(0x02) + chr(0x5F) + chr(0x03) + chr(0xEF) +
chr(0x50))) # Configure IOState Register command

time.sleep(0.050) # Wait 50 ms

# Set the software enable and set the power indicator levels OFF
ser.write((chr(0x4F) + chr((OnOff & 0x40)) + chr(0x50))) # Write IOState
Register command

# Set the laser current setpoint
ser.write((chr(0x53) + chr(0x9C) + chr(0x02) + chr(0x00) + chr(bitset_I) +
chr(0x50))) # Write laser current setpoint command

# Set the temperature setpoint
ser.write((chr(0x53) + chr(0x30) + chr(0x02) + chr(0x00) + chr(bitset_T) +
chr(0x50))) # Write temperature setpoint command

# Reset hysteresis for power indicator levels (and set the software enable)
ser.write((chr(0x4F) + chr((OnOff & 0x40)) + chr(0x50))) # Write IOState
Register command; 1st step: set low level for all power indicator levels
ser.write((chr(0x4F) + chr(((OnOff & 0x40) + 0xB3)) + chr(0x50))) # Write
IOState Register command; 2nd step: set high level for all power indicator
levels

time.sleep(0.500) # Wait 500 ms

# Read the internal enable, the enable bus, the overtemperature alarm and
the power indicator levels of the IOState Register
ser.write((chr(0x49) + chr(0x50))) # Read IOState Register command
time.sleep(0.050) # Wait 50 ms
IOState_Register = ser.read(1) # Read data byte
print("IOState Register: PL5 | EB | PL3 | PL4 | IE | OS | PL2 | PL1")
print("IOState Register (bin) = 0b" +
"{0:08b}".format(ord(IOState_Register)))

# Serial port
ser.close() # Close serial port
```

