Current LED Driver Programming Utility User Guide

The Current Driver Programming Utility is a 32-bit command line program for Windows which can configure supported Current programmable output drivers using a variety of hardware interfaces.

Table of Contents:

- Current Driver Programming Utility
  - Safety Warning
  - System Requirements
  - QuickStart
  - Basic Operation
  - Examples
    - Command Examples
    - Output Example
- Detailed Manual
  - Supported Hardware Interfaces
    - 0-10V Programming Interface
    - DALI USB Interface
  - Wiring Diagrams
    - 0-10V Programming Interface with 0-10V Driver
    - DALI USB Interface with Programmable DALI Driver
    - DALI USB Interface with Programmable DALI Driver (with Auxiliary output)
  - Parameters
    - Utility Configuration Parameters
    - Driver Identification Parameters
    - Driver Configuration Parameters
  - Exit Codes
  - About the Program

Safety Warning

Programming Current LED drivers involves working with mains voltages and Class 1 output voltages. Always power off the mains voltage supply to the driver when connecting or disconnecting leads. Both input and output voltage leads can have dangerously high voltages; Only training and qualified personnel should perform the wiring and programming of LED drivers.

System Requirements

- Microsoft Windows 10
- A supported hardware interface

QuickStart

1. Install the program by running and following the prompts of the installer, `Current_Driver_Programming_Utility_vX.X.X.exe`
2. Attach the appropriate programming interface for the LED Driver to both the PC and the driver. (See Wiring Diagrams)
3. Safely power on the LED driver.
4. Launch the executable `program_driver` from the shell with the desired configuration for the attached driver. The executable has been added to the path during installation. For example, to program GED36MCC2P700 to have a 600mA output:

```
program_driver --description GED36MCC2P700 --set_max_output_current_ma 600
```

Basic Operation

The Current Driver Programming Utility installs as a command line program, `program_driver`, that is launched from the shell (e.g. `cmd` or `powershell`). The program receives all of the programming configuration as named command line arguments, carries out the driver programming, and exits with an exit code value according to resulting status of the driver programming. The program is not interactive, once launched it immediately begins the driver programming according to the arguments received.

To program a driver, the Current Driver Programming Utility needs:

1. One driver identification parameter. This tells the utility what driver is attached so it can select the correct programming technique for that specific driver.
2. One or more driver configuration parameters. These tell the utility what configuration to send to the attached driver.

These arguments are all specified by name, their order does not matter. To see all of the available command line parameters, refer to Parameters

Examples

Command Examples

- Program driver with product code 34047 to a maximum output current of 600mA:

```
program_driver --product_code 34047 --set_max_output_current_ma 600
```

- Program driver with description GED50MCC/CR1P600 to a maximum output current of 450mA, and set its combo control mode to 0-10V:

```
program_driver --description "GED50MCC/CR1P600" --set_max_output_current_ma 450 --
set_combo_control_mode ZERO_TO_TEN_VOLT
```

- Program driver with description GED100MCC2P480 to:
  - A maximum output current of 404mA
  - A minimum output current of 100mA
  - Set the auxiliary supply to auto-detect (supply mode off, sensing on)
  - Calibrate the output current to the driver's attached load.

```
program_driver --description=GED100MCC2P480 --set_max_output_current_ma=404 --
set_min_output_current_ma=100 --set_ul924_sensing_mode=
OFF --set_aux_supply_mode=OFF --set_aux_sensing_mode=ON --calibrate_output_current
```

Output Example

The driver programming commands will print some useful information as the driver programming is happening. Here is an example of the full output for a maximum output current programming session:

```
PS C:\> program_driver --product_code 34047 --set_max_output_current_ma 600
INFO:driver_programmer_backend.get_interfaces:Discovering programming interfaces...
INFO:driver_programmer_backend.get_interfaces:Discovering programming interfaces complete
INFO:programming_fulfiller:Using interface: DALIUSB Serial No. 00019054
INFO:driver_programmer_backend.dali_driver_programmer.DaliDriverProgrammer:Programming max
output current...
INFO:driver_programmer_backend.dali_driver_programmer.DaliDriverProgrammer:Querying max output
current...
Exiting with exit code 0: SUCCESS
```

Detailed Manual

Supported Hardware Interfaces

0-10V Programming Interface

The 0-10V programming interface is a custom programmer provided by Current. It is powered by USB and attaches to the 0-10V dimming leads of Current's programmable 0-10V LED drivers. The 0-10V programming interface is a USB serial device to the PC.

0-10V Programming Interface

DALI USB Interface

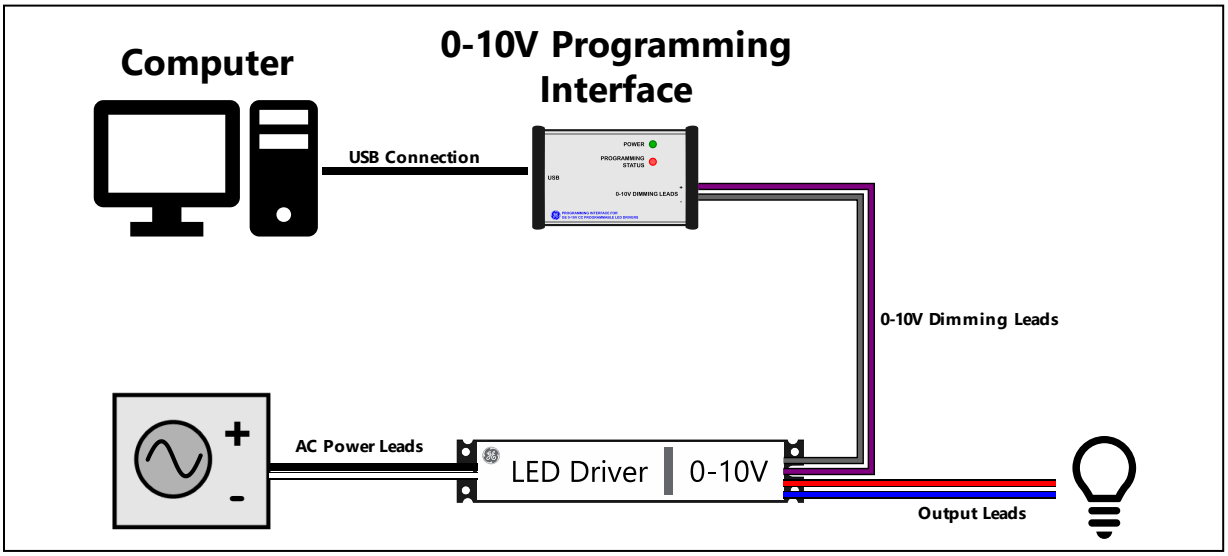
The DALI USB interface is made by Tridonic and provides DALI communication to the PC to which it's attached. It attaches to the DALI leads of Current's programmable DALI or Combo LED drivers. For drivers without an auxiliary supply, a separate DALI bus supply is required to program drivers.

DALI USB Interface

Wiring Diagrams

0-10V Programming Interface with 0-10V Driver

When programming 0-10V drivers, attach the equipment according to the diagram below. The driver must be powered to be programmed.

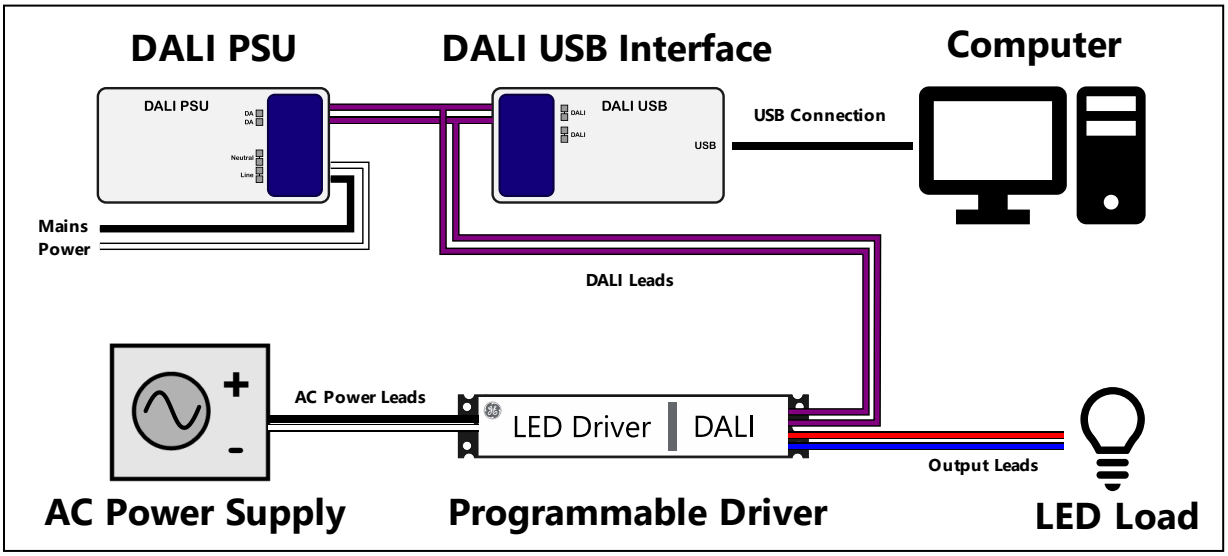


0-10V Programming Interface Wiring Diagram

DALI USB Interface with Programmable DALI Driver

When programming programmable DALI drivers (those that do not have an integrated

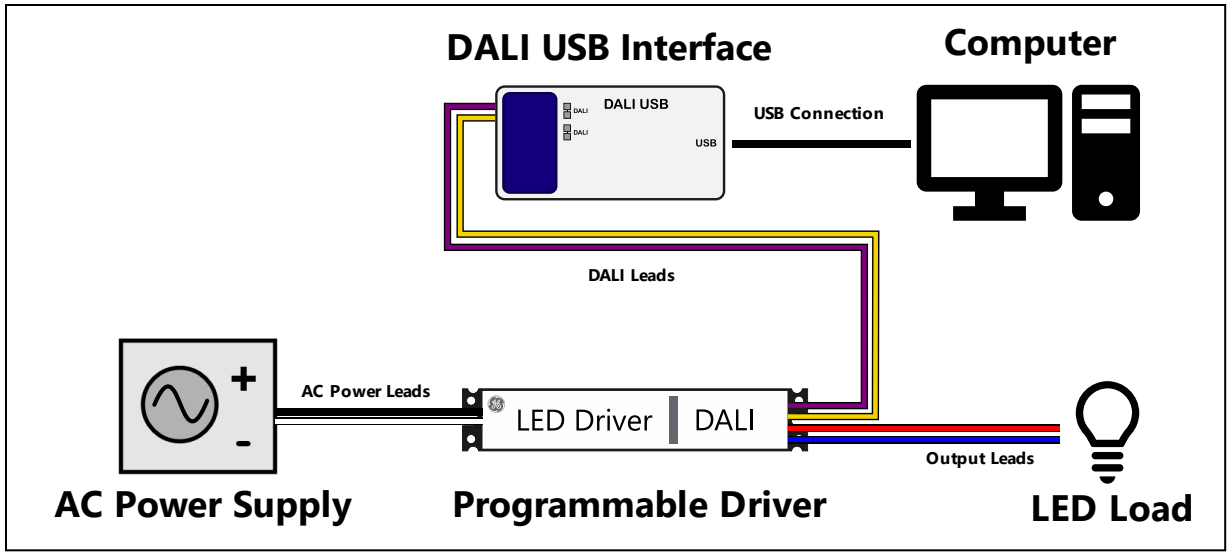
auxiliary supply), attach the equipment according to the diagram below. The driver must be powered to be programmed and the DALI bus must be powered by a DALI bus supply.



DALI USB Interface Wiring Diagram

DALI USB Interface with Programmable DALI Driver (with Auxiliary output)

When programming programmable DALI drivers, attach the equipment according to the diagram below. The driver must be powered to be programmed and the DALI bus is powered by the driver unless it has been configured to turn off its auxiliary supply.



DALI USB Interface Wiring Diagram (with Auxiliary Output Driver)

Parameters

Utility Configuration Parameters

These parameters are for program configuration and are not related to the programming of any specific driver. Most driver programming operations should not require any of these.

-h, --help

Print the help message and exit

--version

Print the program name, version, and compatible driver table, then exit

--print_interface_ids

Find all attached interfaces and print their type and ids

--interface_id INTERFACE_ID

Specify a specific hardware interface by id (see available ids with --print_interface_ids). Without this option, the program will automatically detect and select the appropriate

interface. Use this option to force a specific interface (e.g. there are multiple programming interfaces attached to one PC)

Driver Identification Parameters

These parameters are used to identify a specific driver so that the program can determine the appropriate interface, programming method, and configuration required for that specific model. Generally, only one of these should be provided.

--description DRIVER_DESCRIPTION

Input driver description (e.g. GED36MCC2P700) to identify the attached driver.

--product_code PRODUCT_CODE

Input driver product code (e.g. 503884) to identify the attached driver. The driver's SAP number can also be used here. Either the product code, or the SAP number should be found on the driver's label

--force_programming_method PROG_METHOD

Force a programming method to be used, ignoring provided driver identification and using the method specified. Must be one of `DALI_1`, `DALI_2`, `ZERO_TO_TEN_VOLT_GEN1`, `ZERO_TO_TEN_VOLT_GEN2`.

Driver Configuration Parameters

These parameters are what is used to actually change the configuration of an attached LED driver. Not all drivers support all of these features, the program will exit with an error code if a driver configuration is specified for a driver model that does not support it.

--set_combo_control_mode COMBO_CONTROL_MODE

Set a combo driver to use a specific control mode. Must be one of `DALI`, `ZERO_TO_TEN_VOLT`

--set_max_output_current_ma TARGET_OUTPUT_MILLIAMPS

Set driver maximum output current in miliamps. Must be used with a driver identification parameter.

--set_min_output_current_ma TARGET_OUTPUT_MILLIAMPS

Set driver minimum output current in milliamps. Must be used with a driver identification parameter.

--set_max_output_current_percent TARGET_OUTPUT_PERCENT

Set driver maximum output current in percent (relative to the LED driver's maximum capability).

--set_min_output_current_percent TARGET_OUTPUT_PERCENT

Set driver minimum output current in percent (relative to the LED driver's maximum capability).

--calibrate_output_current

Some LED driver SKUs support calibrating their output current to the attached load for better dimming performance. This switch parameter will perform that calibration on the driver (driver SKU must support it).

--set_aux_supply_mode AUX_SUPPLY_MODE

Set a driver with programmable auxilary to on, off or auto. Note that auto is not supported on all drivers with programmable auxilary supplies. Must be one of `ON`, `OFF`, `AUTO`

--set_ul924_sensing_mode UL924_MODE

Set the UL924 sensing function on the driver. Must be one of `ON`, `OFF`

--set_dim_to_off_mode DIM_TO_OFF_MODE

Set the dim-to-off mode on the driver . Must be one of `ON`, `OFF`

Exit Codes

Value	Name
0	SUCCESS
1	UNCAUGHT_EXCEPTION
2	NO_ARGUMENTS
3	INVALID_ARGUMENTS
4	DRIVER_NO_RESPONSE
5	DRIVER_VALUE_MISMATCH
6	HARDWARE_INTERFACE_NOT_DETECTED
7	HARDWARE_INTERFACE_ERROR

SUCCESS

Programming of the supplied configuration was successful and verified by the driver(if possible with the given interface).

UNCAUGHT_EXCEPTION

The program encountered an internal error. For more information, try again with a lower log-level, e.g. `--log_level DEBUG`

NO_ARGUMENTS

No arguments were supplied to the program.

INVALID_ARGUMENTS

Supplied arguments were invalid or incompatible as a set. Some examples:

- The LED driver could be defined by conflicting identification methods
- A programming method was forced but isn't supported on the driver requested
- The requested driver does not support the configuration requested
- The arguments have syntax errors
- The driver description or product code is not recognized/supported

DRIVER_NO_RESPONSE

The driver did not respond to value confirmation queries. Check that the driver is powered on and connected to the hardware interface. If it's a DALI driver, ensure that the DALI bus is powered.

DRIVER_VALUE_MISMATCH

The driver responded with an unexpected value. E.g. a configuration value was rejected by the driver.

HARDWARE_INTERFACE_NOT_DETECTED

The programming interface hardware is not detected on the PC. Check device manager and verify what interfaces are detected by the program with `--print_interface_ids`

HARDWARE_INTERFACE_ERROR

The hardware interface had some error, e.g. the DALI bus is down or some other problem occurred with the hardware interface

About the Program

The Current Driver Programming Utility installs by deploying an embedded copy of Python to the target machine. This copy of Python does not interfere with any programs, or other versions of Python on the system.

The installation includes a *driver data file* which informs the program of each supported LED driver and its configuration bounds. The Current Driver Programming Utility is not intended to guide the user to what configurations are available for certain driver models, nor will it offer guidance for what configurations are needed for a specific light fixture. It is intended only to provide an automatable interface for manufacturing facilities to easily program Current LED drivers according to parameters that have been validated by an engineering design team.