XMC1200

Microcontroller Series for Industrial Applications

RGB LED Lighting Shield with XMC1202 for Arduino

- ✓ Introduction
- ✓ Board Description
- ✓ Getting Started
- ✓ I²C Master-Slave Communication
- ✓ Programming a master Arduino board to control the RGB LED Lighting Shield
- ✓ Setting the Parameters for YOUR LED Lamp

Board Manual

V1.0 2014-11

Microcontrollers

Edition 2014-11

Published by
Infineon Technologies AG
81726 Munich, Germany

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

Revision History

Page or Item	Subjects (major changes since previous revision)
V1.0, 2014-11	

Table of Contents

About this document

Scope and purpose

This document describes how to use the RGB LED Shield with XMC1202 for Arduino.

Intended audience

Engineers, hobbyists and students who want to add flicker-free LED control to Arduino projects.

Related information

Table 1 Supplementary links and document references

Reference	Description
XMC Microcontrollers	32-bit Industrial Microcontroller based on ARM® Cortex™-M from Infineon
XMC1000 Reference Manuals	Documents section contains reference information for XMC1000 microcontrollers
XMC Development Support	XMC Development Tools
Arduino Home Page	All information on Arduino
Arduino Uno Product Page	Arduino Uno R3 description
Infineon Arduino Page	Boards offered by Infineon for Arduino
DAVE™ Development Platform	All details on DAVE™ IDE
J-Link Debug Probes Product Page	Contains information on J-Link Debug Probes

RGB LED Lighting Shield Introduction

1 Introduction

The RGB LED Lighting Shield adds brilliant flicker-free light control to Arduino projects. The Shield communicates with a master board via the I^2C protocol as a slave. Either an Arduino Uno R3 or the XMC1100 Boot Kit from Infineon can be used as the master.

On board the RGB LED Shield is an XMC1202 microcontroller, featuring a dimming control peripheral for LED lighting applications, known as the Brightness and Colour Control Unit (BCCU). It contains 3 independent dimming engines and 9 independent Pulse Density Modulated (PDM) channels. 1 dimming engine and 6 channels are used in this shield.

There are 10 basic sets of I^2C commands to control the shield from the master board, and so control the connected LED Lamp with various lighting effects. There are 22 user configurable parameters and the freedom to connect different LED Lamps.

The RGB LED Lighting Shield can be easily connected to any Arduino board or the XMC1100 boot Kit via headers and DMX512 control is enabled as a mounting option using an interface chip.

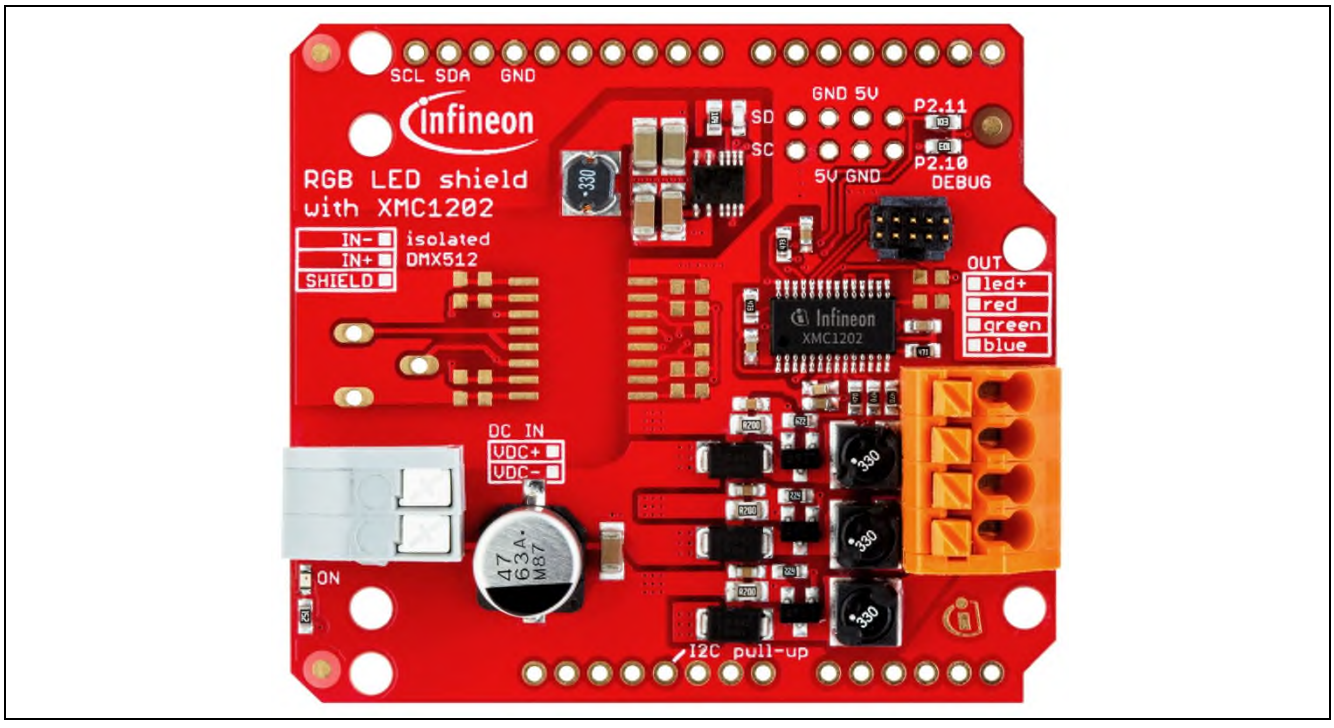


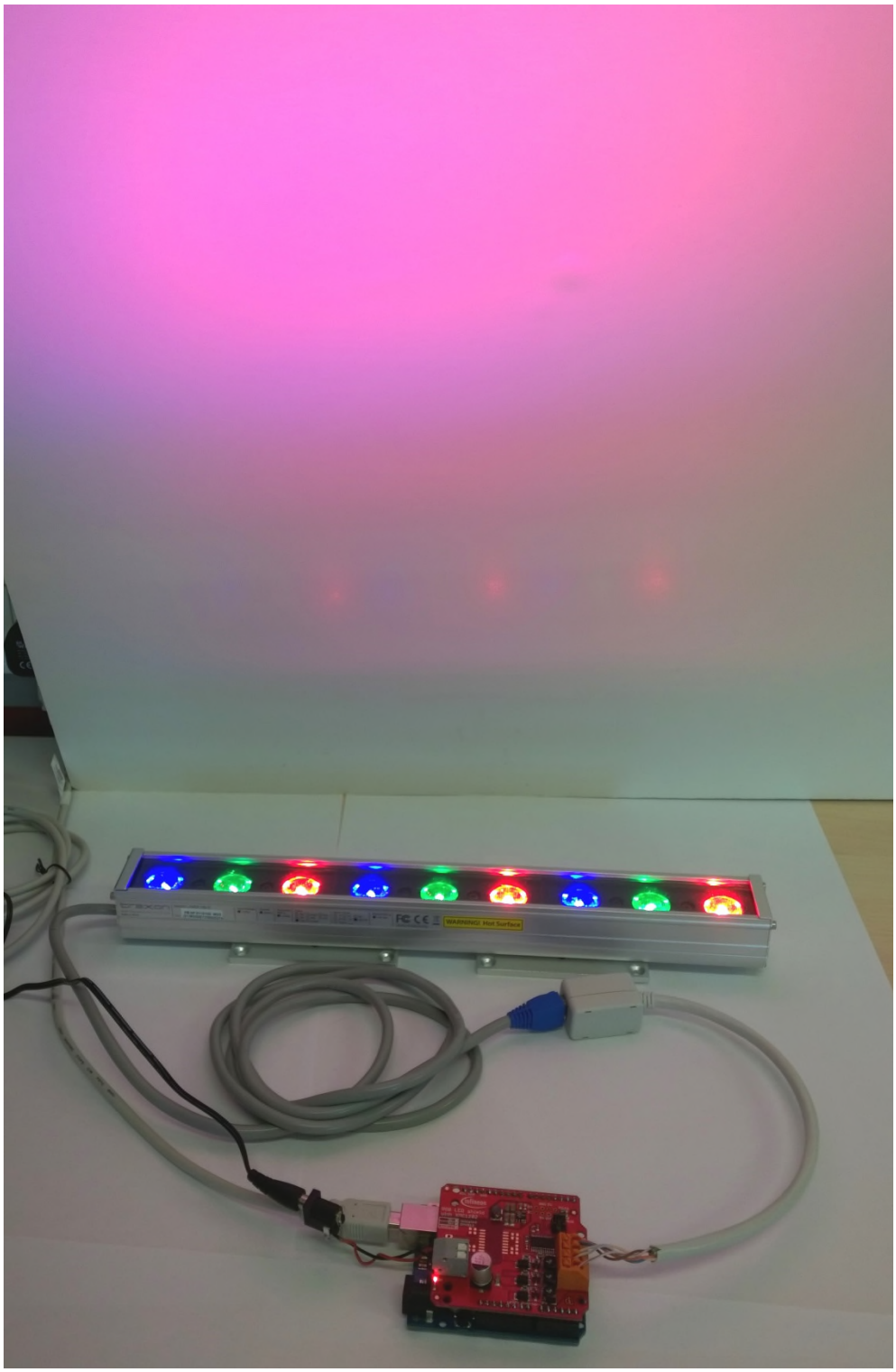
Figure 1 RGB LED Shield photo

1.1 Key Features

The RGB LED Shield has the following features:

- Behaves as an I^2C slave.
  - An Arduino Uno R3, XMC1100 Boot Kit, or similar board connected to the shield can communicate via the SDA and SCL pins as the master.
- Drives and dims up to 3 LED strings with constant current.
- Able to change the colour of a connected LED lamp(if the strings are of different colours; for example red, green, blue).
- High speed flicker-free modulation dimming on each string with Pulse-Density Modulation (PDM).
- Very high power density due to high switching frequency, leading to a small area.

- Up to $48V_{DC}$ input.
  - The RGB LED Shield is a DC-DC buck LED driver so the input voltage must be higher than the forward voltage of the LED strings.
- Configurable current amplitude.
- Up to 700mA average current on each string.
- Configurable current ripple.
- I²C interface with configurable 10-bit slave address (with a default value of 0x15E) to increases the range of devices that can be connected to the bus line.



RGB LED Shield driving an LED wall washer

1.2 Key Features of the XMC1200 MCU series

- 32-bit ARM® Cortex™-M0, 32MHz.
- Hardware Interconnect Matrix.
- 16kB ~ 200kB Flash with ECC and 16kB RAM.
- Peripherals running up to 64MHz.
- Timer/PWM: CCU4, CCU8, POSIF.
- Analog-mixed Signal: 12-bit ADCs, 12-bit DACs, ACMPs.
- Communication: I^2C, SPI, Dual-/Quad-SPI, SCI, I2S, LIN.
- Application specific: LED Color Control Engine, Touch.
- AES 128-bit secure loader for SW IP protection.
- Operating: 1.8 ~ 5.5Volt and -40° ~ 105°C.
- Free DAVE™ IDP and DAVE Apps (code library) open to 3rd party tools and the wide ARM® ecosystem.

1.3 Getting started

The RGB LED Shield uses high frequency peak-current control with fixed off-times to generate DC LED currents. Although this is highly efficient, low cost, and is suitable for high-speed dimming, it results in the output current being dependent on the input and output voltage ratio. The output current can be adjusted by configuring the peak-current reference and off-time parameters.

A virgin RGB LED Shield is pre-configured with safe peak-current reference and off-time parameters. With the safe parameters, the LED current will not be 'too high' at high input voltages.

The safe parameter values have been tested with LED loads that have a forward voltage of 6V at input voltages up to 48V. At this input, the pre-configured average LED current is measured up to 300mA.

Note: LED strings that have a forward voltage lower than 6V and current capability lower than 300mA should not be connected without re-configuring the shield first.

The safe parameters will however result in a discontinuous current with most LED strings and input voltages. For low-ripple continuous current, the off-time and peak-current reference parameters must be configured by the user once the LED lamp and input voltages have been selected.

Generally, the current in any of the strings should never exceed 1A, the peak-current reference parameters should be kept below 0x80, and off-time parameters should be kept above 0x10.

Attention: Improper configuration may result in permanent damage.

RGB LED Lighting Shield Board Description

2 Board Description

The RGB LED Shield can be controlled by programming a master Arduino board, such as the Arduino Uno R3 or the XMC1100 Boot Kit.

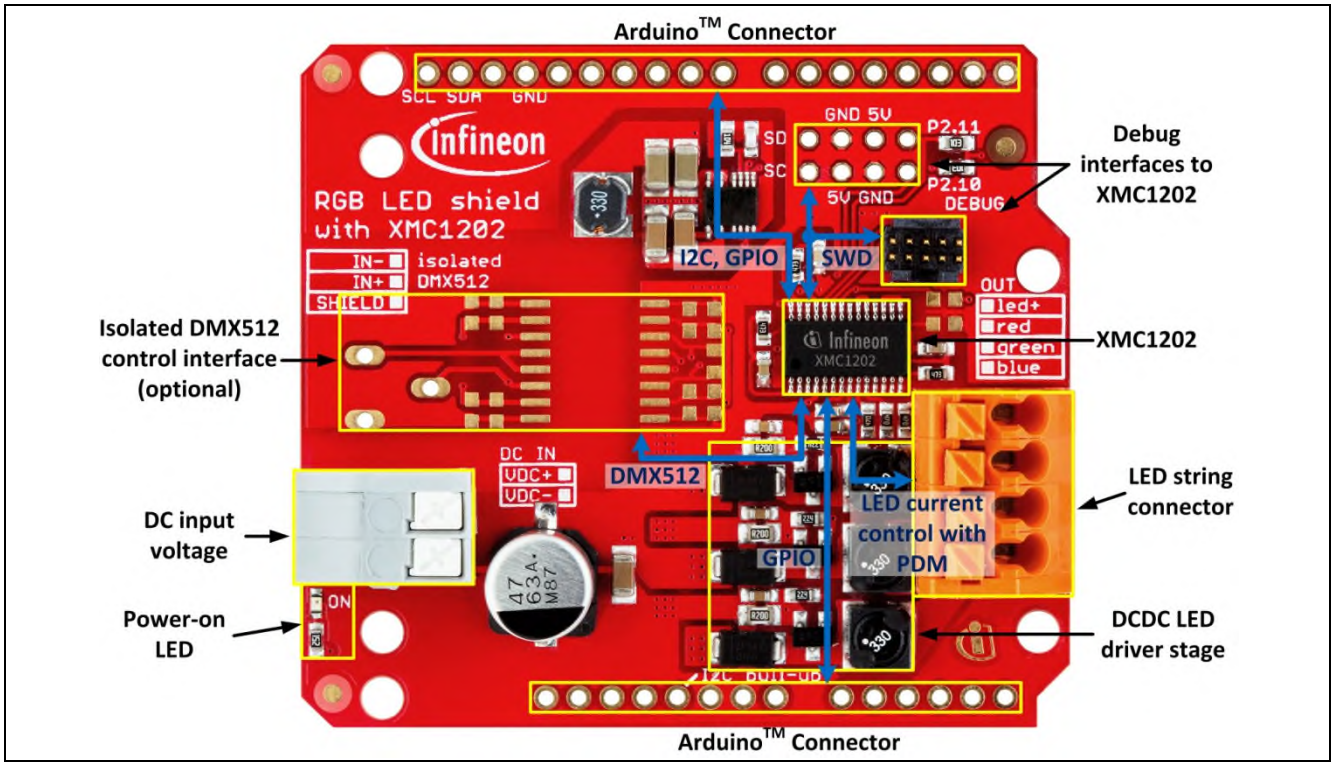


Figure 2 RGB LED Shield Interfaces

2.1 Specifications

Dimensions	2.7 x 2.1 inches (standard Arduino footprint)
Input voltage	up to 48V
Output Current per string	up to 1A peak and 700mA average
Order Number	KIT_LED_XMC1202_AS_01

2.2 Programming Access

The on-board XMC1202 microcontroller can be programmed over SWD via the debug interfaces using a J-Link debug probe from Segger that supports ARM® Cortex™-M0 (Figure 3).

Flash content can be updated over SWD using the TASKING debugger integrated in DAVE™.

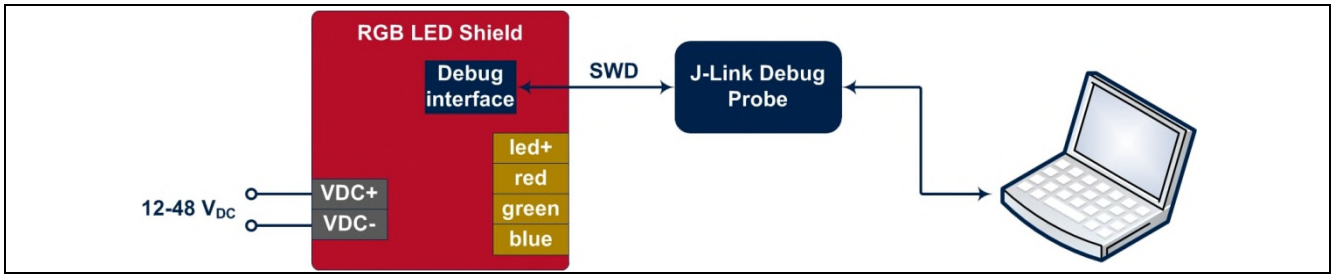


Figure 3 Segger J-Link debug probe connected to the RGB LED Shield

2.3 Schematics and Layout

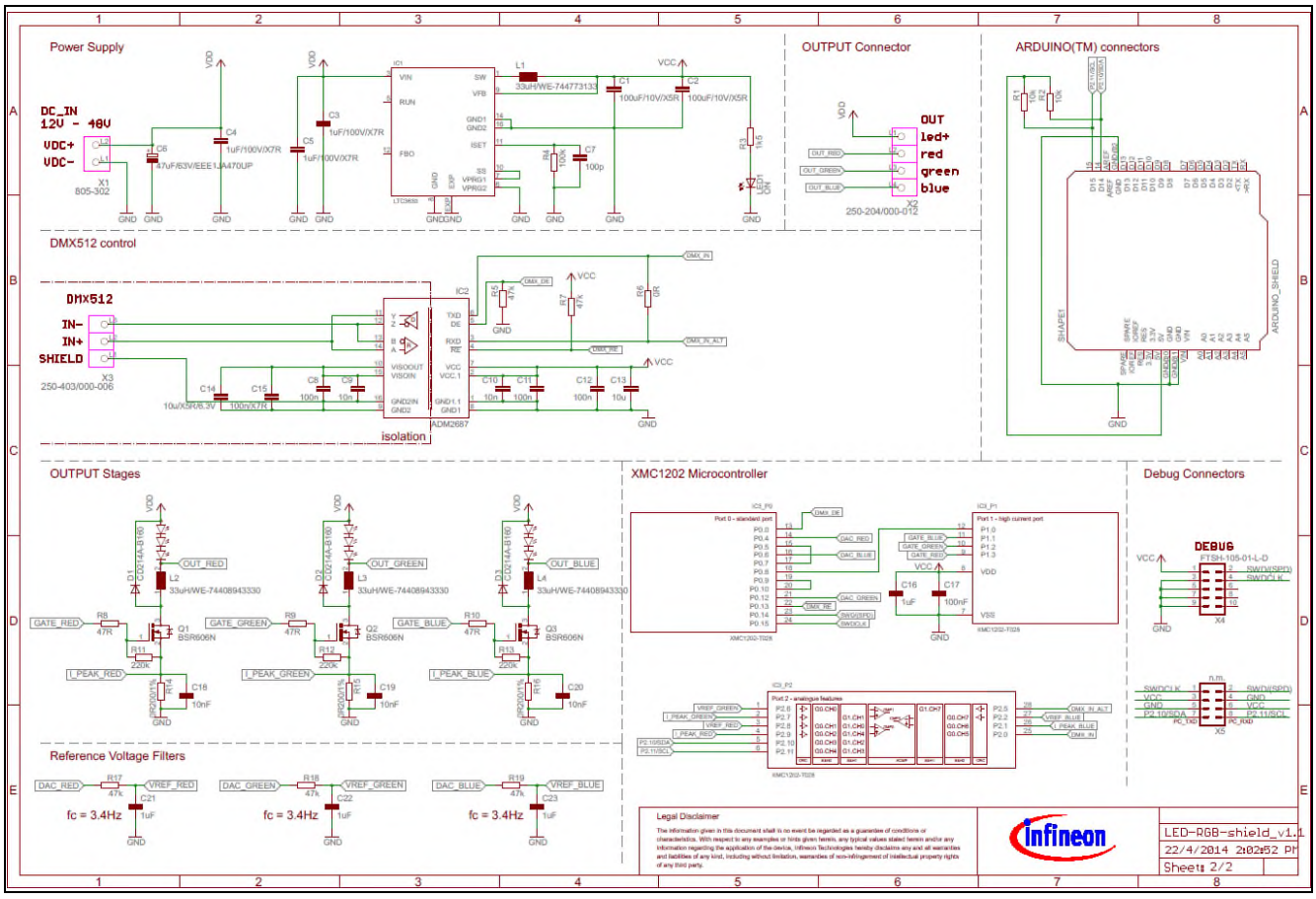


Figure 4 RGB LED Shield – Schematics

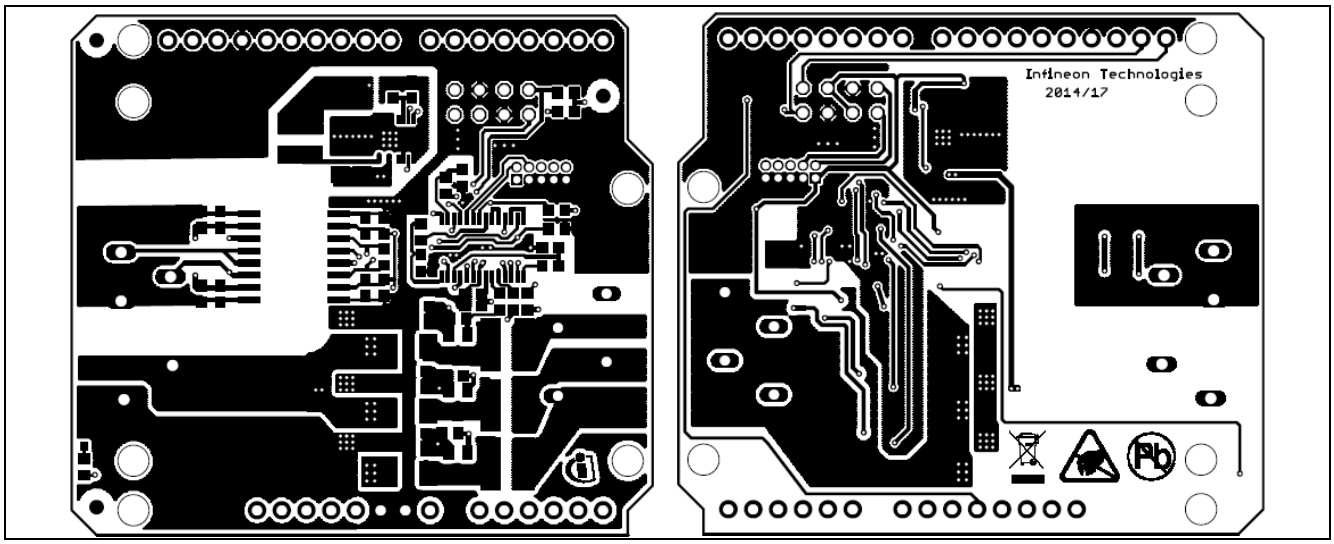


Figure 5 RGB LED Shield – Top and Bottom Layers

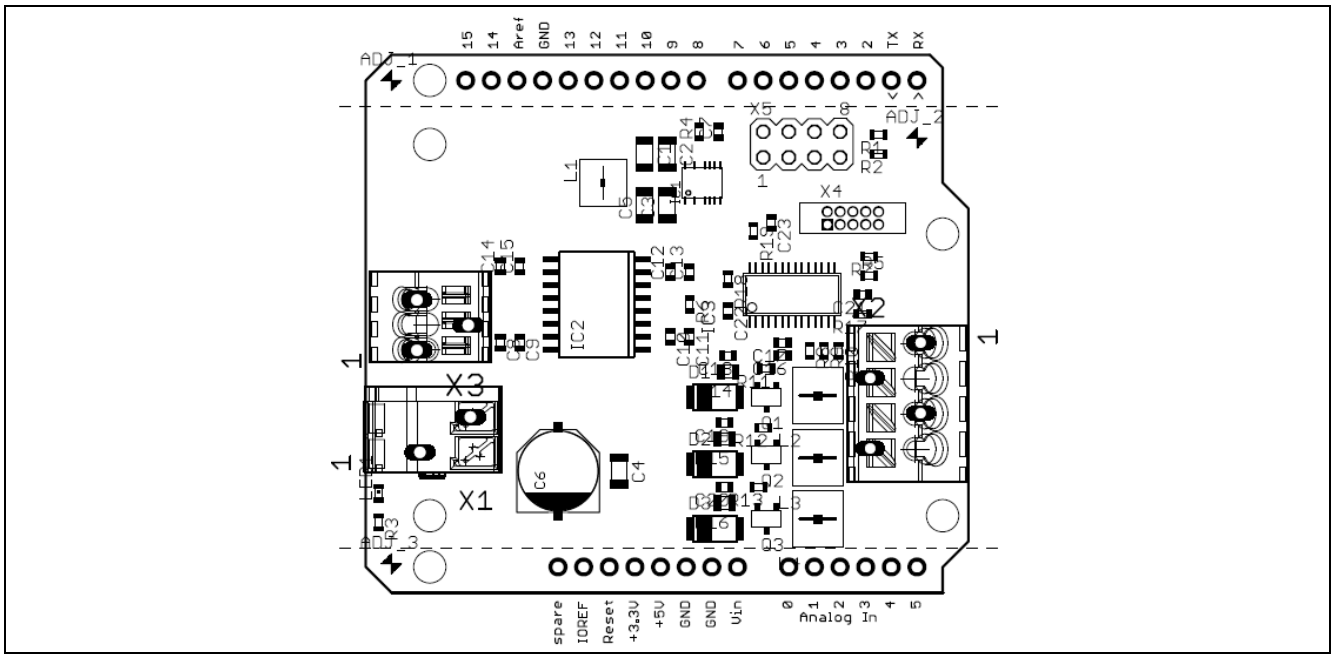


Figure 6 RGB LED Shield – Components

Partlist

Exported from LED-RGB-shield_v1.1.brd at 22/4/2014 2:05:49 PM

Updated on 23/5/2014

EAGLE Version 5.7.0 Copyright (c) 1988-2010 CadSoft

Part	Value	Package	Library	Position (mil)	Orientation	Comment	MPN
C1	100uF/10V/X5R	C1206	rcl	(1087.5 1712.5)	R270	-	TDK C3216X5R1A107M160AC
C2	100uF/10V/X5R	C1206	rcl	(1175 1712.5)	R270	-	TDK C3216X5R1A107M160AC
C3	1uF/100V/X7R	C1206	rcl	(1175 1512.5)	R90	-	AVX 12061C105K4Z2A
C4	1uF/100V/X7R	C1206	rcl	(987.5 475)	R270	-	AVX 12061C105K4Z2A
C5	1uF/100V/X7R	C1206	rcl	(1087.5 1512.5)	R90	-	AVX 12061C105K4Z2A
C6	47uF/63V/EEE1JA470UP	PANASONIC_F	rcl	(750 475)	R90	-	Panasonic EEE1JA470UP
C7	100p	C0603	rcl	(1375 1800)	R90	-	Multicomp MC0603N101J500CT
C8	100n	C0603	rcl	(525 975)	R270	DMX512	AVX 06033C104JAT2A
C9	10n	C0603	rcl	(600 975)	R270	DMX512	Multicomp MC0603B103K160CT
C10	10n	C0603	rcl	(1187.5 1000)	R270	DMX512	Multicomp MC0603B103K160CT
C11	100n	C0603	rcl	(1262.5 1000)	R270	DMX512	AVX 06033C104JAT2A
C12	100n	C0603	rcl	(1187.5 1250)	R90	DMX512	AVX 06033C104JAT2A
C13	10u	C0603	rcl	(1262.5 1250)	R90	DMX512	Murata GRM188R60J106ME47D
C14	10u/X5R/6.3V	C0603	rcl	(525 1275)	R90	DMX512	Murata GRM188R60J106ME47D
C15	100n/X7R	C0603	rcl	(600 1275)	R90	DMX512	AVX 06033C104JAT2A
C16	1uF	C0603	rcl	(1625 925)	R180	-	Murata GRM188R60J105KA01D
C17	100nF	C0603	rcl	(1625 975)	R180	-	AVX 06033C104JAT2A
C18	10nF	C0603	rcl	(1412.5 925)	R180	-	Multicomp MC0603B103K160CT
C19	10nF	C0603	rcl	(1400 662.5)	R180	-	Multicomp MC0603B103K160CT
C20	10nF	C0603	rcl	(1400 412.5)	R180	-	Multicomp MC0603B103K160CT
C21	1uF	C0603	rcl	(1937.5 1162.5)	R180	-	Murata GRM188R60J105KA01D
C22	1uF	C0603	rcl	(1412.5 1100)	R270	-	Murata GRM188R60J105KA01D
C23	1uF	C0603	rcl	(1581.252 1443.752)	R270	-	Murata GRM188R60J105KA01D
D1	CD214A-B160	DO-214AC	diode	(1362.5 762.5)	R0	-	Bourns CD214A-B160LF
D2	CD214A-B160	DO-214AC	diode	(1362.5 500)	R0	-	Bourns CD214A-B160LF
D3	CD214A-B160	DO-214AC	diode	(1362.5 250)	R0	-	Bourns CD214A-B160LF
IC1	LTC3630	MSE16(12)	LinearTechnology	(1312.5 1600)	R0	-	Linear Technology LTC3630EMSE#PBF
IC2	ADM2687	SO-16W	AnalogDevice	(900 1125)	R90	DMX512	Analog Devices ADM2687EBRIZ
IC3	XMC1202-T028	P-TSSOP28-16	XMC1000	(1662.5 1162.5)	R90	-	Infineon Technologies XMC1202-T028X0016 AA
L1	33uH/WE-744773133	WE-PD2_S	wuerth-elektronik	(925 1600)	R270	-	Wuerth Elektronik WE-744773133
L2	33uH/WE-74408943330	WE-SPC_48XX	PowerMagnetics	(1762.5 768.752)	R0	-	Wuerth Elektronik WE-74408943330
L3	33uH/WE-74408943330	WE-SPC_48XX	PowerMagnetics	(1762.5 525)	R0	-	Wuerth Elektronik WE-74408943330
L4	33uH/WE-74408943330	WE-SPC_48XX	PowerMagnetics	(1762.5 287.5)	R0	-	Wuerth Elektronik WE-74408943330
LED1	ON	CHIP-LED0603	led	(50 387.5)	R0	-	Wuerth Elektronik 150060RS75000
Q1	BSR606N	SC59	infineon_discrete	(1562.5 762.5)	R180	-	Infineon Technologies BSR606N
Q2	BSR606N	SC59	infineon_discrete	(1562.5 537.5)	R180	-	Infineon Technologies BSR606N
Q3	BSR606N	SC59	infineon_discrete	(1562.5 287.496)	R180	-	Infineon Technologies BSR606N
R1	10k	R0603	rcl	(2000 1787.5)	R180	-	Bourns CR0603-JW-103ELF
R2	10k	R0603	rcl	(2000 1712.5)	R180	-	Bourns CR0603-JW-103ELF
R3	1k5	R0603	rcl	(50 275)	R270	-	Bourns CR0603-JW-152GLF
R4	100k	R0603	rcl	(1300 1800)	R90	-	Bourns CR0603-JW-104ELF
R5	47k	R0603	rcl	(1962.5 1237.5)	R0	DMX512	Bourns CR0603-JW-473GLF
R6	0R	R0603	rcl	(1262.5 1125)	R270	DMX512	Bourns CR0603-J/-000ELF
R7	47k	R0603	rcl	(1962.5 1312.5)	R180	DMX512	Bourns CR0603-JW-473GLF
R8	47R	R0603	rcl	(1731.252 943.752)	R270	-	Bourns CR0603-JW-470GLF
R9	47R	R0603	rcl	(1787.5 943.752)	R270	-	Bourns CR0603-JW-470GLF
R10	47R	R0603	rcl	(1843.752 943.748)	R270	-	Bourns CR0603-JW-470GLF
R11	220k	R0603	rcl	(1562.5 875)	R180	-	Bourns CR0603-FX-2203ELF
R12	220k	R0603	rcl	(1550 643.752)	R180	-	Bourns CR0603-FX-2203ELF
R13	220k	R0603	rcl	(1531.252 412.5)	R180	-	Bourns CR0603-FX-2203ELF
R14	0R200/1%	R0805	rcl	(1412.5 862.5)	R180	-	Bourns CRL0805-FW-R200ELF
R15	0R200/1%	R0805	rcl	(1400 600)	R180	-	Bourns CRL0805-FW-R200ELF
R16	0R200/1%	R0805	rcl	(1400 350)	R180	-	Bourns CRL0805-FW-R200ELF
R17	47k	R0603	rcl	(1937.5 1087.5)	R180	-	Bourns CR0603-JW-473GLF
R18	47k	R0603	rcl	(1412.5 1225)	R270	-	Bourns CR0603-JW-473GLF
R19	47k	R0603	rcl	(1512.5 1412.5)	R270	-	Bourns CR0603-JW-473GLF
SHAPE1	ARDUINO_SHIELD	XMC1100_FOR_ARDUINO	arduino	(0 0)	R0	-	
X1	805-302	P-805-302	con-wago_805	(262.5 625)	R180	-	WAGO 805-302
X2	250-204/000-012	P-250-204	con-wago_250	(2112.5 737.5)	R0	-	WAGO 250-204/000-012
X3	250-403/000-006	P-250-403	con-wago_250	(250 1075)	R180	DMX512	WAGO 250-403/000-006
X4	FTSH-105-01-L-D	FTSH-105-01-L-DV-THROUGHHOLE	TIMM1	(1800 1437.5)	R0	-	Samtec FTSH-105-01-L-DV
X5	n.m.	MA04-2	con-lstb	(1700 1750)	R0	-	Multicomp 10-89-7082

Figure 7 RGB LED Shield – BOM

Getting Started

3 Getting Started

You can bring YOUR LED lamp to life in seven simple steps.

STEP 1. Choose a high-power light engine

a. Maximum three channels (e.g. RGB)

b. Minimum 300mA LED current rating

NOTE: If the current rating is <300mA you can easily configure your RGB LED Lighting Shield using the instructions in chapter 6 (Parameter Setup for YOUR LED Lamp).

c. Maximum 48V forward voltage per LED channel

STEP 2. Choose a DC adapter

a. Input voltage to the RGB LED Lighting Shield: 12V ~ 48V DC

b. Maximum 48V forward voltage per LED channel

NOTE: DC input voltage to the RGB LED Lighting Shield should be higher than the forward voltage of the LED channels.

STEP 3. Solder pin headers on the RGB LED Lighting Shield

STEP 4. Connect the RGB LED Lighting Shield to

a. Arduino Uno R3

b. XMC1100 Boot Kit

STEP 5. Program Arduino Uno R3 or XMC1100 Boot Kit

a. Example Sketches and projects: www.infineon.com/arduino

i. Upload *RGBLED_2_SAFE.ino* to Arduino Uno R3

ii. Upload *RGBLED_2_Safe_XMC11.zip* to XMC1100 Boot Kit

STEP 6. Connect the DC adapter to the RGB LED Lighting Shield

STEP 7. Turn on the power

I²C Master-Slave Communication Protocol

4 I²C Master-Slave Communication Protocol

Command words have been defined in software. Parameters can be changed by sending these commands from the master to the RGB LED Lighting Shield. These commands can be sent to the shield from the master board using pre-defined functions.

4.1 Brief Description of I²C Functions

The I²C commands together with the required data can be sent to the RGB LED Lighting Shield from the master board using the functions provided. These functions encapsulate the data in the necessary format for transfer via the I²C communication protocol.

The functions are provided for the Arduino Uno R3 and the XMC1100 Boot Kit.

The RGB LED Shield's I²C address is a 10-bit address and is pre-configured to be 0x15E. To address it, the master will send 2 bytes of address:

- The first 7 bits of the first byte are 11110XX, of which XX are the two most significant bytes of the 10-bit address. The 8th bit determines the read or write direction of the data transfer.
- The second byte is the lower 8-bits of the address.

Write functions

I2CWRITE2BYTES, I2CWRITE6BYTES, I2CWRITE9BYTES, I2CWRITE_DIRECTACCESS, I2CCHANGEADDRESS, I2CDMX and I2CSAVEPARAM

- The I²C START condition is sent, followed by the 1st byte of the RGB LED Shield address byte, a 'zero' bit to indicate a transmission request and the 2nd address byte.
- The appropriate command word is then sent, followed by the data and a STOP condition to terminate the transfer. Data is always put on the SDA line as a byte that is 8-bits long. 16-bit data is sent as 2 bytes and 32-bit data as 4 bytes.

Read functions

I2CREAD, I2CREAD_DIRECTACCESS

- The I²C START condition is sent, followed by the 1st byte of the RGB LED Shield address byte, a 'zero' bit to indicate a transmission request and the 2nd address byte.
- The appropriate command word is then sent.
- A repeated START condition is then sent followed by the 1st byte of the RGB LED Shield address byte, a 'zero' bit, the 2nd address byte and the 1st address byte with a 'one' bit to request for data.
- Acknowledge pulses are subsequently sent.
- A STOP condition is sent to terminate the transfer.

Note: A detailed description of each function can be found in the Appendix.

2.1.1 Command Overview Table

The following tables provides a short description of the commands that can be sent with the functions.

Table 2 Commands and Functions

I^2C Commands	Description	I^2C Function used
INTENSITY_RED	Change relative colour intensity of red channel	I2CWRITE2BYTES
INTENSITY_GREEN	Change relative colour intensity of green channel	I2CWRITE2BYTES
INTENSITY_BLUE	Change relative colour intensity of blue channel	I2CWRITE2BYTES
INTENSITY_RGB	Change relative colour intensity of red, green and blue channels	I2CWRITE6BYTES
CURRENT_RED	Change peak-current reference of red channel	I2CWRITE2BYTES
CURRENT_GREEN	Change peak-current reference of green channel	I2CWRITE2BYTES
CURRENT_BLUE	Change peak-current reference of blue channel	I2CWRITE2BYTES
OFFTIME_RED	Change off-time of red channel	I2CWRITE2BYTES
OFFTIME_GREEN	Change off-time of green channel	I2CWRITE2BYTES
OFFTIME_BLUE	Change off-time of blue channel	I2CWRITE2BYTES
WALKTIME	Change walktime of red, green and blue channels	I2CWRITE2BYTES
DIMMINGLEVEL	Change brightness level	I2CWRITE2BYTES
FADERATE	Change time taken to dim to 0%	I2CWRITE2BYTES
CHANGEADDRESS	Change address of RGB LED Shield	I2CWRITE2BYTES
DMXOFF	Disable DMX512 control	I2CDMX
DMXON	Enable DMX512 control	I2CDMX
DMXSLOT	Change first relevant slot of DMX512 control	I2CWRITE2BYTES
DMX8BIT	Read 8-bits of colour information from each DMX512 slot	I2CWRITE6BYTES
DMX16BIT	Read 16-bits of colour information from each DMX512 slot	I2CWRITE12BYTES
READ_CONFIG	Query if RGB LED Shield has been configured	I2CREAD
READ_INTENSITY_RED	Request for relative colour intensity of red channel	I2CREAD
READ_INTENSITY_GREEN	Request for relative colour intensity of green channel	I2CREAD
READ_INTENSITY_BLUE	Request for relative colour intensity of blue channel	I2CREAD
READ_CURRENT_RED	Request for peak current reference of red channel	I2CREAD
READ_CURRENT_GREEN	Request for peak current reference of green channel	I2CREAD
READ_CURRENT_BLUE	Request for peak current reference of blue channel	I2CREAD
READ_OFFTIME_RED	Request for off-time of red channel	I2CREAD
READ_OFFTIME_GREEN	Request for off-time of green channel	I2CREAD
READ_OFFTIME_BLUE	Request for off-time of blue channel	I2CREAD
READ_WALKTIME	Request for linear walk time	I2CREAD
READ_DIMMINGLEVEL	Request for dimming level	I2CREAD
READ_FADERATE	Request for rate of dimming	I2CREAD
READ_DMX	Query if DMX512 control is enabled	I2CREAD
READ_DMXSLOT	Request for first relevant slot in DMX512 control	I2CREAD
READ_DMXBIT	Request for number of bits of colour information expected from DMX512 control	I2CREAD
READ_DMXREDH	Request for slot which stores upper 8-bits of red colour information	I2CREAD
READ_DMXREDL	Request for slot which stores lower 8-bits of red colour information	I2CREAD

I²C Commands	Description	I²C Function used
READ_DMXGREENH	Request for slot which stores upper 8-bits of green colour information	I2CREAD
READ_DMXGREENL	Request for slot which stores lower 8-bits of green colour information	I2CREAD
READ_DMXBLUEH	Request for slot which stores upper 8-bits of blue colour information	I2CREAD
READ_DMXBLUEL	Request for slot which stores lower 8-bits of blue colour information	I2CREAD
DIRECTACCESS_READ	Request for value contained in a specific register	I2CREAD_DIRECTACCESS
DIRECTACCESS_MOVE	Move value into a specific register	I2CWRITE_DIRECTACCESS
DIRECTACCESS_AND	Bitwise AND operation on a user specified value and the value in a specific register	I2CWRITE_DIRECTACCESS
DIRECTACCESS_OR	Bitwise OR operation on a user specified value and the value in a specific register	I2CWRITE_DIRECTACCESS
SAVEPARAMETERS	Save current parameters to Flash memory	I2CSAVEPARAM

4.2 Command Description

4.2.1 Colour Intensity (INTENSITY_RED, INTENSITY_GREEN, INTENSITY_BLUE, INTENSITY_RGB)

The colour intensities of the Red, Green and Blue colour channels on the RGB LED Lighting Shield can be changed.

Three of the 9 available BCCU channels on the XMC1202 microcontroller on-board the RGB LED Shield are used to control the colour intensities. A change in the relative colour intensity in any of three channels will change the colour of the lamp attached to the shield. Colour intensities are 12-bit values. The maximum intensity of each channel is 0xFFF.

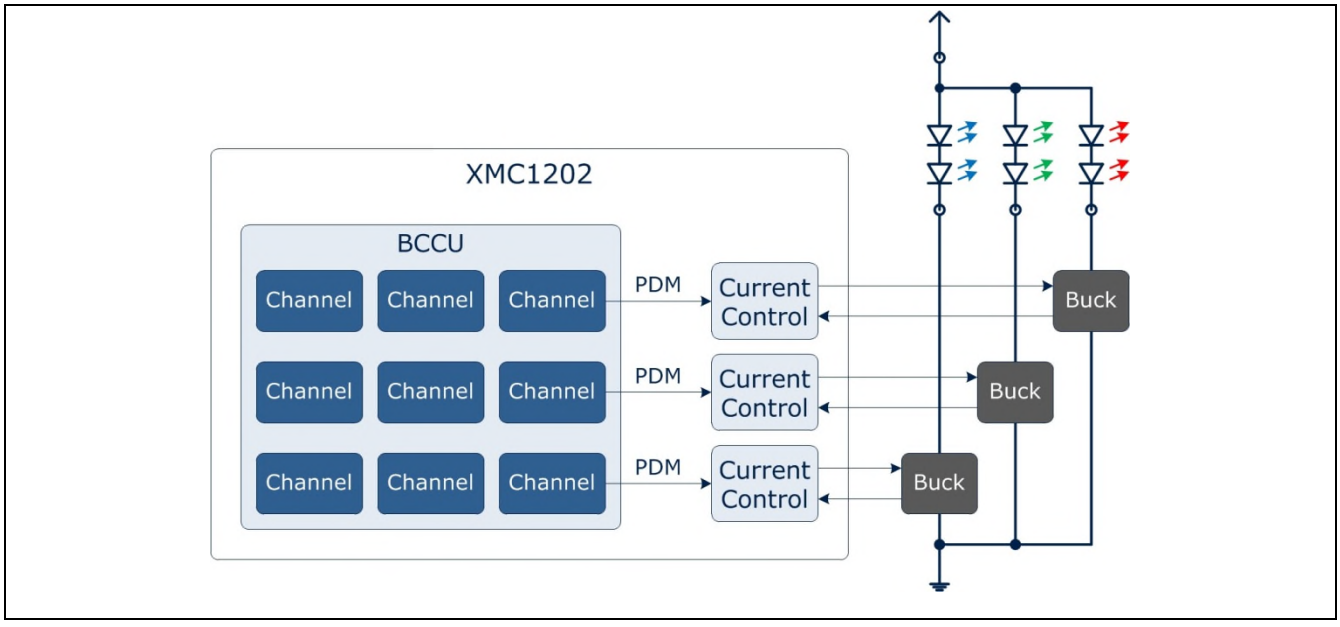


Figure 8 PDM Channels in the microcontroller on board the RGB LED Shield

INTENSITY_RED

Changes the relative colour intensity of the red channel.

To set the red channel to maximum intensity, send the following from the master:

```
I2CWRITE2BYTES(INTENSITY_RED, 0xFFF);
```

INTENSITY_GREEN

Changes the relative colour intensity of the green channel.

To set the green channel to maximum intensity, send the following from the master:

```
I2CWRITE2BYTES(INTENSITY_GREEN, 0xFFF);
```

INTENSITY_BLUE

Changes the relative colour intensity of the blue channel.

To set the blue channel to maximum intensity, send the following from the master

```
I2CWRITE2BYTES(INTENSITY_BLUE, 0xFFF);
```

INTENSITY_RGB

Changes the relative colour intensities of the red, green and blue channel.

To enable white light, send the following from the master

```
I2CWRITE2BYTES(INTENSITY_RGB, 0xFFF);
```

Recommended Colour Scheme

To ensure constant lamp brightness for different colors, keep the sum of intensities of the three channels constant.

Table 3

Colour	Channel Intensity			Possible commands to be sent from the master
	Red	Green	Blue	

Red	0xFFF	0x000	0x000	`I2CWRITE2BYTES(ADDRESS, INTENSITY_RED, 0xFFF);`
				`I2CWRITE2BYTES(ADDRESS, INTENSITY_GREEN, 0x000);`
				`I2CWRITE2BYTES(ADDRESS, INTENSITY_BLUE, 0x000);`
				OR
				`I2CWRITE6BYTES(ADDRESS, INTENSITY_RGB, 0xFFF, 0x000, 0x000);`

Green	0x000	0xFFF	0x000	`I2CWRITE2BYTES(ADDRESS, INTENSITY_RED, 0x000);`
				`I2CWRITE2BYTES(ADDRESS, INTENSITY_GREEN, 0xFFF);`
				`I2CWRITE2BYTES(ADDRESS, INTENSITY_BLUE, 0x000);`
				OR
				`I2CWRITE6BYTES(ADDRESS, INTENSITY_RGB, 0x000, 0xFFF, 0x000);`

Blue	0x000	0x000	0xFFF	`I2CWRITE2BYTES(ADDRESS, INTENSITY_RED, 0x000);`
				`I2CWRITE2BYTES(ADDRESS, INTENSITY_GREEN, 0x000);`
				`I2CWRITE2BYTES(ADDRESS, INTENSITY_BLUE, 0xFFF);`
				OR
				`I2CWRITE6BYTES(ADDRESS, INTENSITY_RGB, 0x000, 0x000, 0xFFF);`

Colour	Channel Intensity			Possible commands to be sent from the master
	Red	Green	Blue	
Yellow	0x800	0x800	0x000	`I2CWRITE6BYTES(ADDRESS, INTENSITY_RGB, 0x800, 0x800, 0x000)`
Cyan	0x000	0x800	0x800	`I2CWRITE6BYTES(ADDRESS, INTENSITY_RGB, 0x000, 0x800, 0x800)`
Magenta	0x800	0x000	0x800	`I2CWRITE6BYTES(ADDRESS, INTENSITY_RGB, 0x800, 0x000, 0x800)`
White	0x555	0x555	0x555	`I2CWRITE6BYTES(ADDRESS, INTENSITY_RGB, 0x555, 0x555, 0x555)`

4.2.2 Peak Current Reference (CURRENT_RED, CURRENT_GREEN, CURRENT_BLUE)

The LED current can be controlled by the RGB LED shield. When attached to the shield, the LED lamp is connected to a 3-channel DCDC buck LED driver.

An inductor, Schottky diode and MOSFET are used, in an inverted buck topology, to control the LED current with high efficiency. As with every DC-DC buck driver, this design results in ripples in the LED current. In the RGB LED shield, the ripple frequency is approximately 1-1.5 MHz to support fast modulation dimming and achieve high power density.

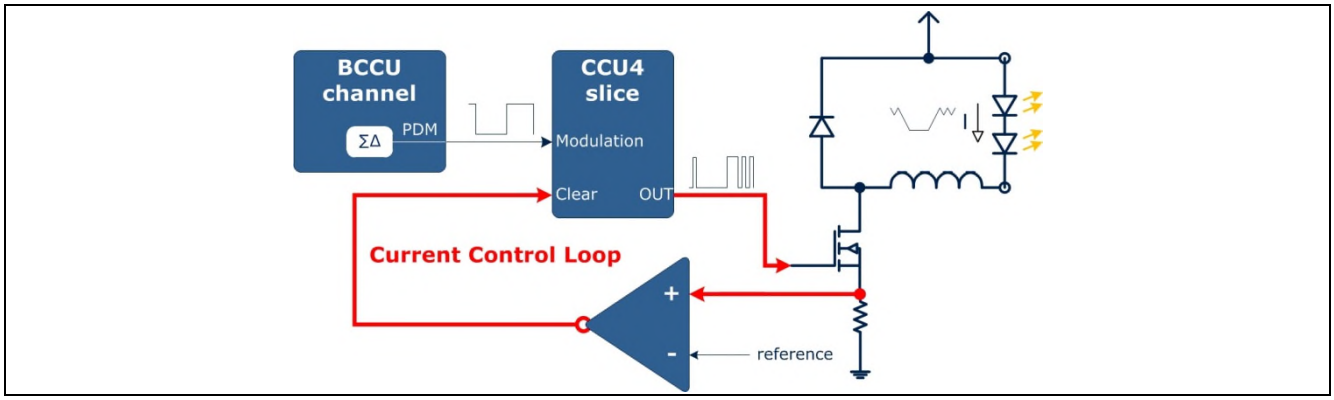


Figure 9 Peak Current Control

To adjust the LED current, the potential before the shunt resistor is fed into an on-chip comparator. The inductor in the setup causes the LED current to increase linearly and proportionately to the input voltage. As the current increases, the potential before the shunt resistor increases. When this potential exceeds the peak current reference value, the MOSFET is switched off by the MOSFET control output signal which switches to 0V. Current will continue to flow through the free-wheeling diode as the inductor's magnetic field collapses. During this time, the current decreases linearly and proportionately to the forward voltage of the LED string. The process restarts when the MOSFET is switched on after a fixed off-time.

The RGB LED Shield will change the peak current reference parameter when the CURRENT_RED, CURRENT_GREEN or CURRENT_BLUE commands and the 12-bit peak-current reference parameter are sent from the master. A reference value of 0xFFF corresponds to 5V, and 0x000 corresponds to 0V.

To calculate the reference voltage, use:

Reference Value / 4096 * 5V

The maximum peak current reference value that can be set is 0x80, which is approximately 0.15625V. Should a value greater than this be sent to the RGB LED Shield, the value will be ignored and peak

current reference set to 0x80. This corresponds to a theoretical peak current of 781mA flowing through the MOSFET.

CURRENT_RED

Changes the peak current reference parameter of the red channel.

To change the reference value to approximately 0.12V, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, CURRENT_RED, 0x64); // 0.12 = 100 / 4096 * 5
```

CURRENT_GREEN

Changes the peak current reference parameter of the green channel.

To change the reference value to approximately 0.12V, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, CURRENT_GREEN, 0x64);
```

CURRENT_BLUE

Changes the peak current reference parameter of the blue channel.

To change the reference value to approximately 0.12V, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, CURRENT_BLUE, 0x64);
```

4.2.3 Off-Time (OFFTIME_RED, OFFTIME_GREEN, OFFTIME_BLUE)

This parameter adjusts the ripple of the LED current.

When the comparator in the shield detects that the current in the lamp has reached the peak current reference, the MOSFET is switched off. This off-state is extended for a fixed duration determined by the off-time parameter value. In this off-state, the circuit is switched off and the LED current decreases.

The smaller the off-time value, the shorter the off-state, the less the LED current decreases, leading to a valley current which is closer in value to the peak current. As a result, the ripple in the current is reduced.

Conversely, when the off-state is extended for a longer duration, the LED current falls more, resulting in a smaller valley current, and a larger ripple.

Ideally, the LED current should not exceed the peak-current reference.

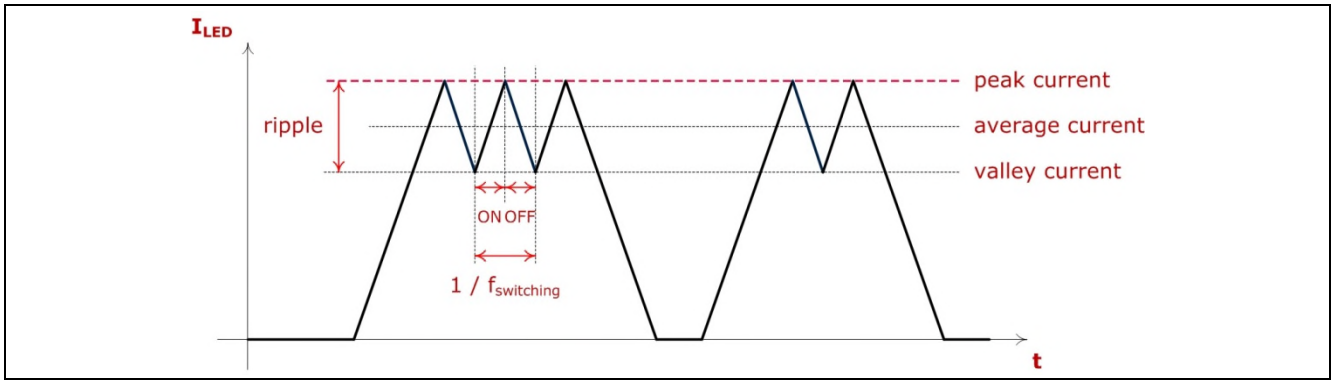


Figure 10 LED Current Ripple

Due to non-negligible propagation delays in the comparator and the connected on-chip circuits, the LED current peaks invariably exceed the peak-current reference. There is a time delay between the LED current reaching the peak-current reference value and the comparator detecting it. A short off-state can result in the current not dropping enough before the MOSFET is switched on again. The comparator may no longer be able to accurately detect the peak current reference, leading to

exceedingly high currents. To avoid catastrophically high currents, the off-state is generated after the LED current has dropped below the peak reference level.

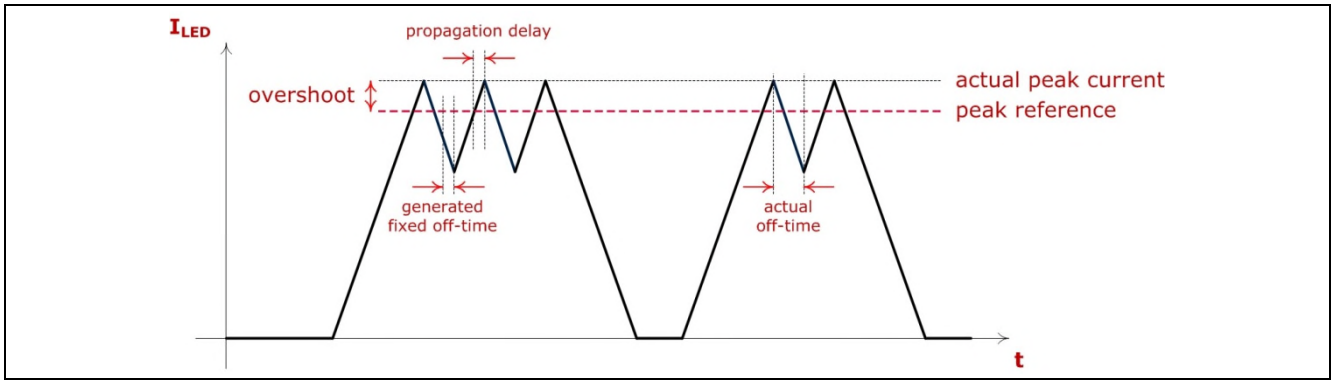


Figure 11 Propagation Delay leading to LED Current Over-shooting Peak Current Reference

The MOSFET will remain off while a counter counts up to the off-time parameter value. When the off-time value is reached, the counter resets and the MOSFET is switched on.

The counter counts at a frequency of 64MHz (resolution of 15.625ns).

The circuit will be in the generated off-state for 1µs when the off-time value is set to:

0x40 (1 / 64M * 64)

OFFTIME_RED

Changes the fixed off-time parameter of the red channel.

To change the off-time to 1µs, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, OFFTIME_RED, 0x40);
```

OFFTIME_GREEN

Changes the fixed off-time parameter of the green channel.

To change the off-time to 1µs, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, OFFTIME_GREEN, 0x40);
```

OFFTIME_BLUE

Changes the fixed off-time parameter of the blue channel.

To change the off-time to 1µs, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, OFFTIME_BLUE, 0x40);
```

4.2.4 Walk time (WALKTIME)

A linear walk is used to smoothly change the colour intensities. The intensities change linearly over time. The time taken for the channels to reach their target intensities is called the linear walk time. The linear walk time can be adjusted.

The RGB LED Shield calculates the actual linear walk time with the formula:

Linear Walk Time = WALKTIME * 0.01024

A WALKTIME value of 0x10 means that the actual linear walk time is 164ms. The channels will take 164ms to reach their target intensities.

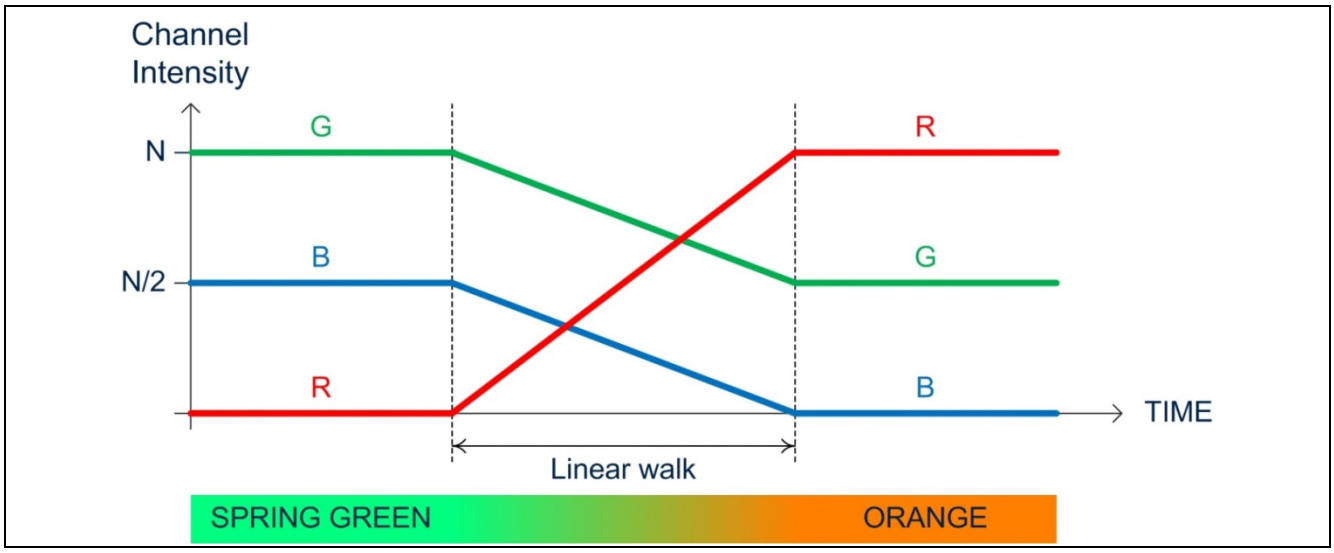


Figure 12 Walk time – Time taken for channels to reach their target intensities

WALKTIME

This command can only be used to change the WALKTIME parameter for all three channels together.

To change the linear walk time to 164ms, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, WALKTIME, 0x10);
```

4.2.5 Dimming (DIMMINGLEVEL)

The dimming engine in the XMC1202 microcontroller on-board the RGB LED Shield performs dimming along a pseudo-exponential curve so that the change appears natural to the human eye. This compensates for the eye's logarithmic sensitivity to light.

The curve is quantized into 4095 steps, giving the user 4096 dimming levels to choose from.

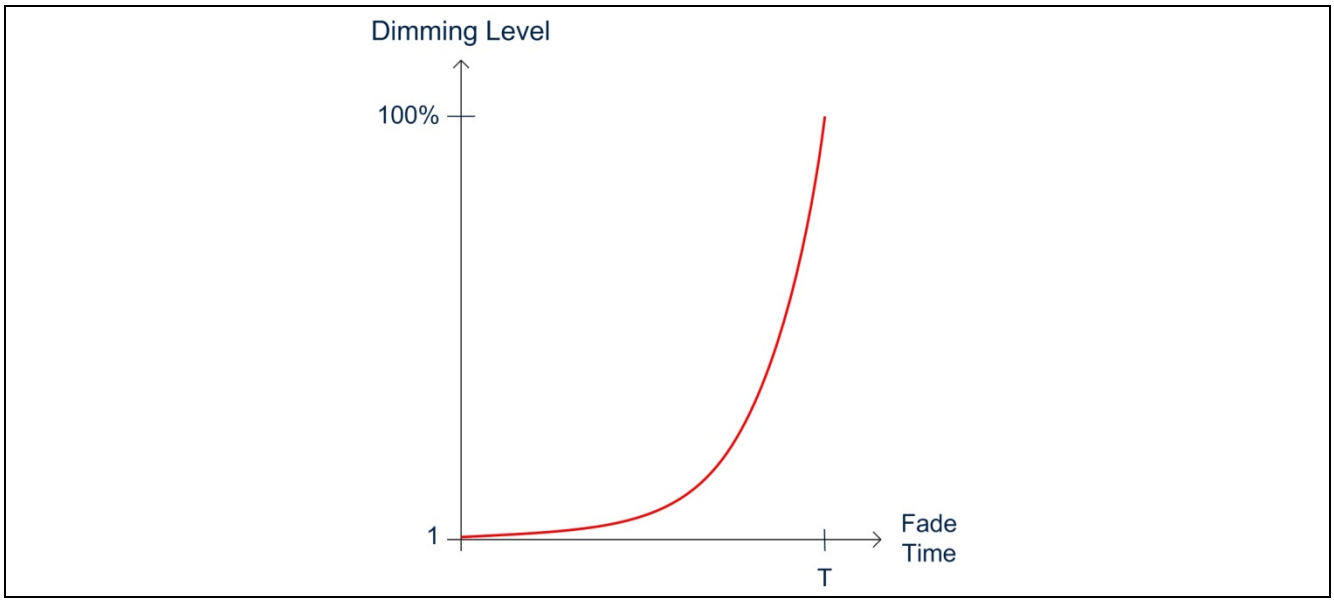


Figure 13 Exponential dimming curve

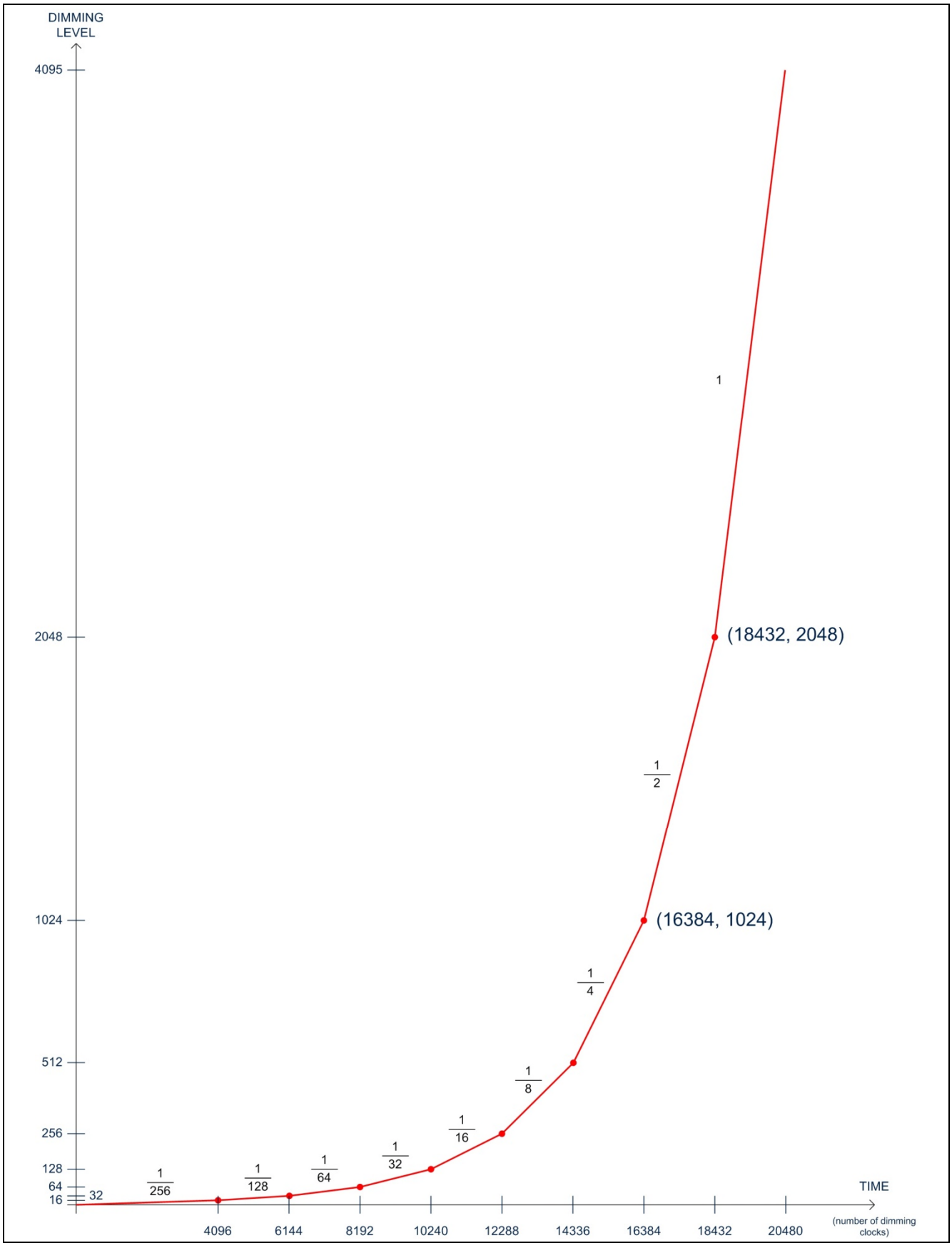


Figure 14 Piece-wise exponential dimming curve

The brightness value of a channel and therefore the brightness of the connected LED string, is the product of the intensity of the respective channel and the dimming level divided by 4096.

The dimming engine which controls the dimming level is separate from the BCCU channels. This enables the RGB LED shield to control the colour of the LED strings separately from the lamp brightness. The brightness level of the lamp can stay the same while its colour changes. Changes in brightness do not affect the colour of the lamp either.

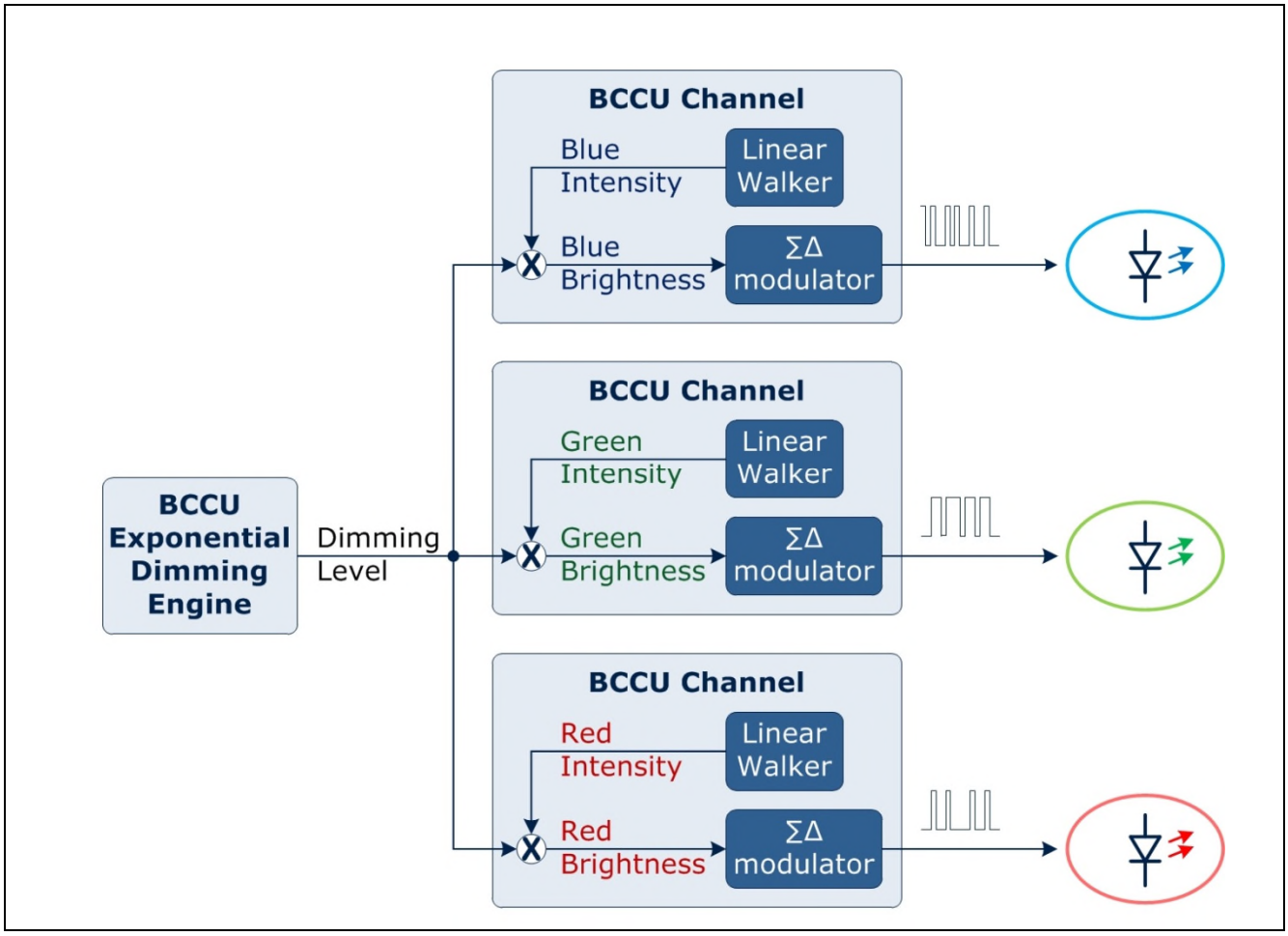


Figure 15 Dimming and Colour Control

Dimming and Colour Control examples

If the lamp is to have green colour at 50% brightness, the colour intensity of the green channel can remain at 4096. The 50% brightness level is achieved by changing the dimming level to 2048 (50%).

Table 4 Green colour at 50% brightness

Channel	Lamp Dimming Level	Channel Intensity	Channel Brightness
RED	2048	0	0
GREEN		4096	2048
BLUE		0	0

If the lamp colour is to be changed to yellow colour at 50% brightness, the dimming level can remain the same at 2048, while only the relative colour intensity changes. The overall lamp brightness remains constant despite changes in colour.

Table 5 Yellow colour at 50% brightness

Channel	Lamp Dimming Level	Channel Intensity	Channel Brightness
RED	2048	2048	1024
GREEN		2048	1024
BLUE		0	0

The lamp colour can be changed to cyan at 25% brightness for example. The dimming level is set to 1024 (25%) and the intensities are changed following the recommended colour scheme.

Table 6 Cyan at 25% brightness

Channel	Lamp Dimming Level	Channel Intensity	Channel Brightness
RED	1024	0	0
GREEN		2048	512
BLUE		2048	512

The brightness level of the lamp is changed by modulation dimming in the RGB LED shield. The brightness value in each LED string is converted to a PDM ON-OFF signal by a sigma-delta modulator, which turns the string on and off fast. The switching rate of this signal is high to avoid flicker. The switching rate is however still lower than the ripple frequency of the MOSFET control signal for current control. The higher the brightness value, the longer the signal from the sigma-delta modulator is ON for. For example, if a 50% brightness value is expected, the signal will be ON for 50% of the time. If a 70% brightness value is expected, the signal will be ON for 70% of the time, and so on. Ideally, the current should have a square waveform. However, the switching circuit results in a ripple in the LED current.

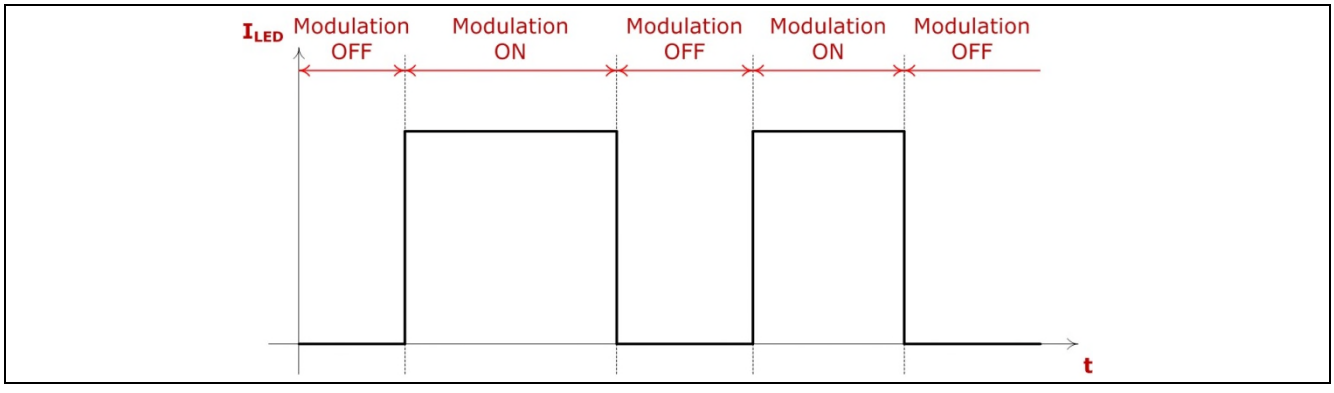


Figure 16 Ideal Modulation Dimming with ideal current control

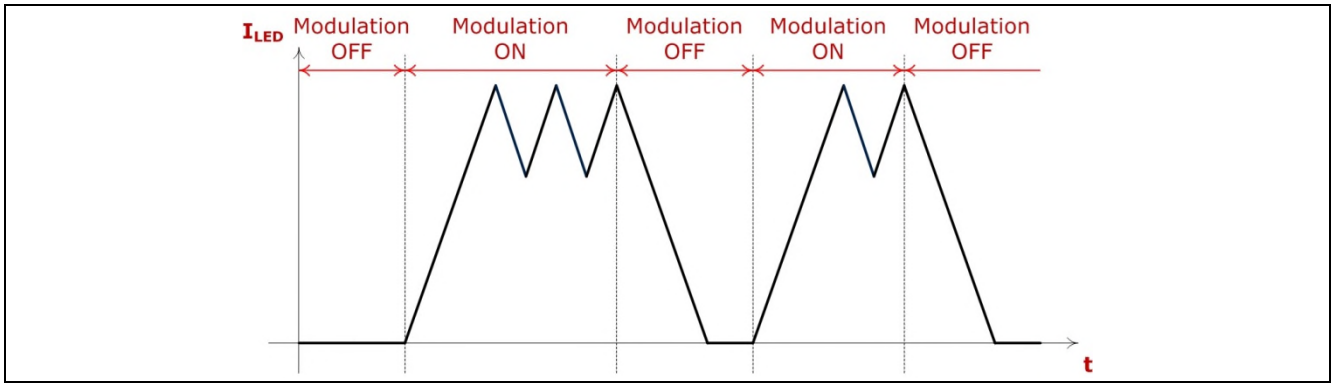


Figure 17 Actual Modulation Dimming with DC-DC current control

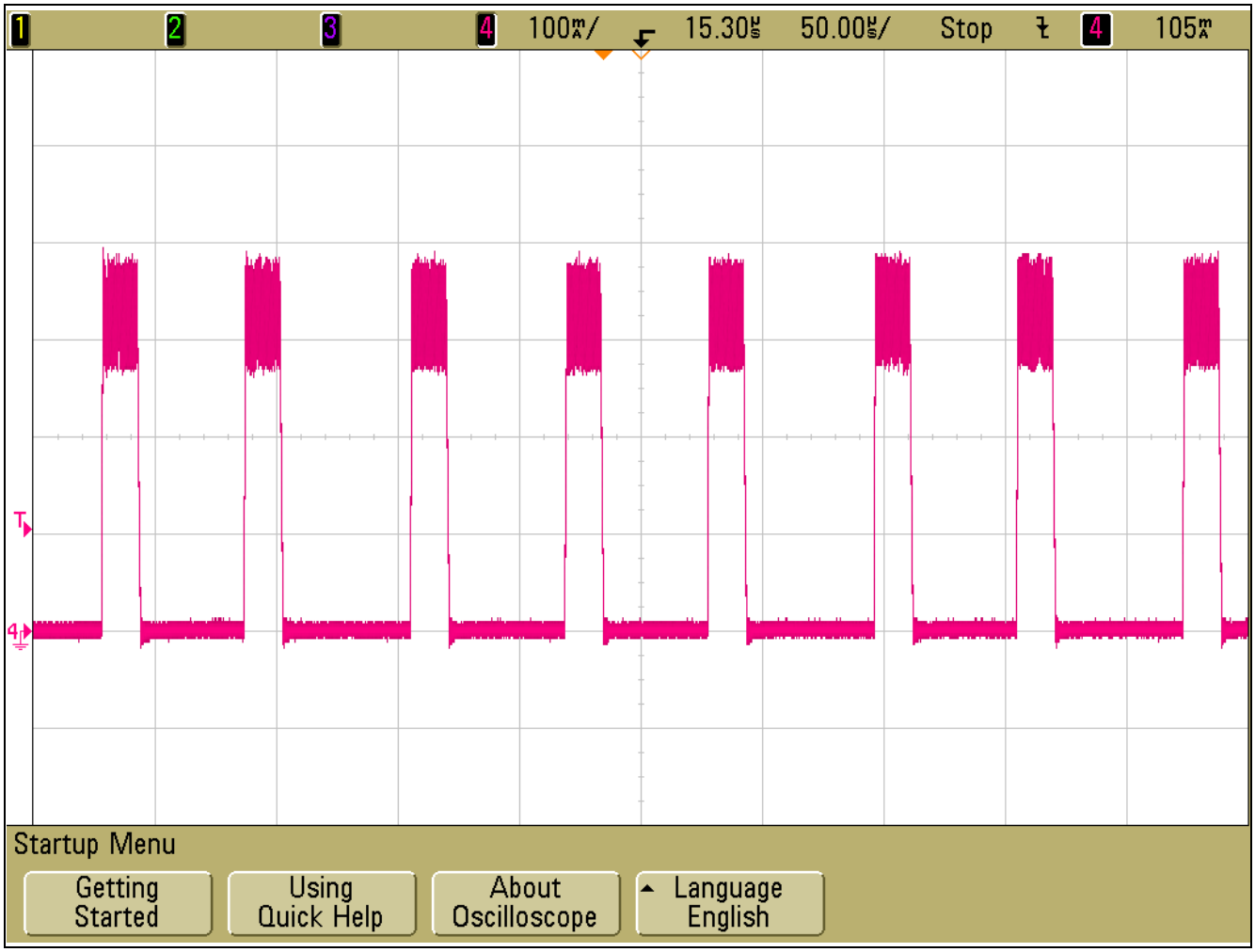


Figure 18 Typical LED Current Waveform with Pulse-Density Modulation (~30% brightness)

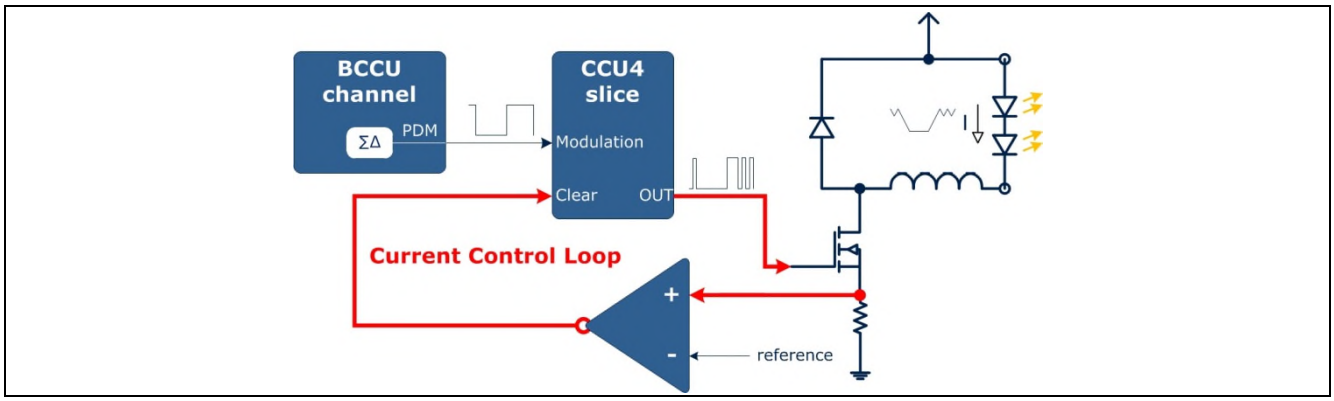


Figure 19 Peak Current Control

DIMMINGLEVEL

Changes the Dimming Level as controlled by the Dimming Engine.

To change the dimming level to 50%, send the following from the master:

```
I2CWRITE2BYTES(ADDRESS, DIMMINGLEVEL, 0x7FF); // or 2048 in place of 0x7FF
```

4.2.6 Fade Rate (FADERATE)

The RGB LED Shield can adjust how fast the lamp can change its brightness value. This is adjusted through the 10-bit FADERATE parameter. The time taken for the brightness to change also depends. Based on Figure 14, the fade time can be calculated:

Fade Time $_{0 \rightarrow 100\%}$

- It takes 20479 dimming clocks for the lamp to dim up from 0 to 100% brightness. The dimming clock frequency is 292.237kHz. Hence, 20479 dimming clock last approximately 0.07001s.
- Fade Time = FADERATE * 0.07001s
- A FADERATE value of 0xA leads to a fade time of approximately 700ms.

Fade Time $_{0 \rightarrow 50\%}$

- It takes 18432 dimming clocks for the lamp to dim up from 0 to 50% brightness. This leads to:
- Fade Time = FADERATE * 0.0630721s
- A FADERATE value of 0xA leads to a fade time of approximately 630ms.

The time taken for the lamp to dim from 0 to 100% brightness is not twice the time taken for the lamp to dim from 0 to 50%. This is due to the exponential nature of the dimming curve. The pseudo-exponential curve is used because the human eye perceives brightness logarithmically, resulting in an observed linear change in brightness.

FADERATE

Changes the time the lamp takes to change the dimming level.

To change the fade time to 700ms, send the following to the master:

```
I2CWRITE2BYTES(ADDRESS, DIMMINGLEVEL, 0xA); // or 10 in place of 0xA
```

4.2.7 DMX512 Control Commands

If an RS485 transceiver chip is mounted on the RGB LED Shield, DMX512 related commands can be sent to the shield to change the way it is expected to behave.

DMX OFF / ON

The DMXOFF command disables the DMX512 control of the RGB LED Shield

The DMXON command enables the control.

To disable the command, send the following from the master:

```
I2CDMX(ADDRESS, DMXOFF);
```

To enable the command, send the following from the master:

```
I2CDMX(ADDRESS, DMXON);
```

DMX Starting Slot

The DMXSTART command changes the first relevant slot that the RGB LED Shield uses to read colour information when DMX control is enabled.

To change the first relevant slot to slot 1, send the following from the master:

```
I2CWRITE2BYTES (ADDRESS, DMXSTART, 0x1);
```

DMX8BIT

The DMX8BIT command sets the RGB LED Shield's DMX input to be three bytes. The shield will expect 8-bit colour information to be available in 3 relevant slots, one for each colour. Together with the DMX8BIT command, the slot numbers for red, green and blue in order, have to be sent to the shield.

For example, if the relevant slots in the DMX512 control slot 3, slot 4 and slot 5 (Table 7), and the first relevant slot has been configured to slot 1, send the following to the master:

```
I2CWRITE6BYTES (ADDRESS, DMX8BIT, 0x2, 0x3, 0x4);
// red channel is at offset 2, green at offset 3 and blue at offset 4
```

Table 7

			Red	**Green**	**Blue**	
Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6

DMX16BIT

The DMX16BIT command sets the RGB LED Shield's DMX input to be six 8-bit slots. The shield expects 16-bit colour information to be available in 6 relevant slots, with 2 bytes for each channel. Together with the DMX16BIT command, the slot numbers for red, green and blue in order, has to be sent to the shield.

To set the shield's DMX input configuration to be for 8-bits information, use the I2CWRITE12BYTES function:

For example, if the relevant slots in the DMX512 control slot 3, slot 4, slot 5, slot 6, slot 7, slot 8, and the first relevant slot has been configured to slot 1, send the following to the master:

```
I2CWRITE12BYTES (ADDRESS, DMX8BIT, 0x2, 0x3, 0x4, 0x5, 0x6, 0x8);
// red channel is at offset 2 and 3, green at offset 4 and 5 and blue at offset 6 and 7
```

Table 8

			Red High	Red Low	Green High	Green Low	Blue High	Blue Low	
Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8	Slot 9

4.2.8 Changing the RGB LED Shield's Address (CHANGEADDRESS)

The 10-bit I2C address of the RGB LED Shield can be changed with the CHANGEADDRESS command. The pre-configured address is 0x15E, but if there are address conflicts with other I^2C devices on the bus, the LED Shield's address can be changed.

To change the address of the shield to 0x147 for example, send the following from the master:

```
CHANGEADDRESS (ADDRESS, 0x147);
```

This is not a permanent change in the RGB LED Shield's address. On the next start-up, the shield's address will be that of the previous address again. If the address should be permanently changed, the SAVEPARAMETERS command can be used to re-configure the shield.

4.2.9 Configuring the RGB LED Shield (SAVEPARAMETERS)

The RGB LED Shield can be configured with values for each of these parameters:

- INTENSITY_RED, INTENSITY_GREEN, INTENSITY_BLUE, INTENSITY_RGB
- CURRENT_RED, CURRENT_GREEN, CURRENT_BLUE
- WALKTIME
- FADERATE
- DIMMINGLEVEL
- OFFTIME
- DMXOFF/ON, DMX8BIT/DMX16BIT, DMXSLOT
- ADDRESS

SAVEPARAMETERS

The SAVEPARAMETERS command instructs the RGB LED Shield to save all current values for the parameters to flash memory.

The parameters are saved to a flash page from address 0x10004F000 to 0x10004FFF. On start-up, these values will be used by default.

To save the current parameters, send the following from the master:

```
I2CSAVEPARAM (ADDRESS);
```

4.2.10 Request for Data (I2CREAD commands)

Any parameter that can be written with I^2C commands can be read back from the RGB LED Shield. The requested information is returned as a 16-bit unsigned integer. The following are the commands to be sent:

READ_INTENSITY_RED, READ_INTENSITY_GREEN, READ_INTENSITY_BLUE

- Request for intensity data for red, green and blue channels.

READ_CURRENT_RED, READ_CURRENT_GREEN, READ_CURRENT_BLUE

- Request for peak-current reference data for red, green and blue channels.

READ_WALKTIME

- Request for Walktime.

READ_FADERATE

- Request for Fade Rate.

READ_DIMMINGLEVEL

- Request for dimming level.

READ_OFFTIME_RED, READ_OFFTIME_GREEN, READ_OFFTIME_BLUE

- Request for off-time for red, green and blue channels.

READ_DMX

- Query if DMX is off or on.

READ_DMXBIT

- Query if DMX is set to 8 or 16-bits.

READ_DMXSLOT

- Request for starting relevant slot.

READ_DMXREDH, READ_DMXREDL, READ_DMXGREENH, READ_DMXGREENL, READ_DMXBLUEH, READ_DMXBLUEL

- Request for the relevant slots for red, green and blue channels.
- If DMX is set to 8-bits, READ_DMXREDL, READ_DMXGREENL and READ_DMXBLUEL will return 0 as they are not in use.

READ_CONFIG

- Queries whether the RGB LED Shield has already been configured.
- If the READ_CONFIG command returns a 0, no values have been saved to the flash memory before. All shields shipped have already been configured with safe parameters. This query should never return 0.

To request for the peak current reference value of the red channel, send the following from the master:

```
redcurr = I2CREAD (ADDRESS, READ_CURRENT_RED);
```

Writing this line in the Arduino sketch will command the shield to send back the value of the peak current reference value of the red channel. The I2CREAD function returns the received data, and prints it to the serial monitor as a hexadecimal value. To read the other parameters, replace READ_CURRENT_RED with the appropriate command.

4.2.11 Directly Accessing Registers

Registers in the XMC1202 microcontroller on board the RGB LED Lighting Shield can also be accessed directly without the use of the pre-defined I^2C commands. It is therefore possible to access registers which may not be accessible with the I^2C commands.

*Note: This is an advanced way of accessing registers and is only recommended for experienced users. Detailed information on the XMC1202 microcontroller can be found at **www.infineon.com/XMC1000**.*

DIRECTACCESS_READ

The DIRECTACCESS_READ command is used to read the value in a register.

To read the value contained in a register, send the following from the master:

```
Data = I2CREAD_DIRECTACCESS (SHIELD_ADDRESS, REGISTER_ADDRESS);
```

Entering this line in an Arduino Sketch will return the value in the register at REGISTER_ADDRESS as a 32-bit unsigned integer.

DIRECTACCESS_MOVE, DIRECTACCESS_AND, DIRECTACCESS_OR

These are the 3 commands that can be used to write to a register.

Attention: Ensure that the register being written to is safe to be overwritten. Otherwise, unpredictable behaviour or permanent damage to the RGB LED Shield may occur.

- DIRECTACCESS_MOVE

This command replaces the value in the register specified in REGISTER_ADDRESS, with a different value.

If the replacement value is 0x50 for example, send the following from the master:

```
I2CWRITE_DIRECTACCESS (ADDRESS, DIRECTACCESS_MOVE, REGISTER_ADDRESS, 0x50);
```

- DIRECTACCESS_AND

This command implements a bitwise AND of the contents of a specified value and the current value in REGISTER_ADDRESS.

To perform a bitwise AND on the register's value with 0x11001100, send the following from the master:

```
I2CWRITE_DIRECTACCESS (ADDRESS, DIRECTACCESS_AND, REGISTER_ADDRESS, 0x11001100);
```

If the register value is 0x11110000 for example, a bitwise AND with 0x11001100 would result in a value of 0x11000000.

- DIRECTACCESS_OR

This command implements a bitwise OR of the contents of a specified value with the current value in REGISTER_ADDRESS.

To perform a bitwise OR on the register's value with 0x11005100, send the following from the master:

```
I2CWRITE_DIRECTACCESS (ADDRESS, DIRECTACCESS_AND, REGISTER_ADDRESS, 0x11005100);
```

If the register value is 0x11110000 for example, the OR with 0x11005100 would result in a value of 0x11115100.

Programming a master Arduino board to control the RGB LED Lighting Shield

5 Arduino Compatibility

The RGB LED Shield can be controlled by programming a master Arduino Board.

5.1 Simple Test Program

A simple Arduino sketch with only minimal coding is needed to start the operation of the attached LED lamp.

```
// the loop routine runs over and over again forever:
void loop() {
  // Flash Red for 0.5s
  I2CWRITE2BYTES (ADDRESS, INTENSITY_RED, 0x0FFF); // RED at maximum intensity
  I2CWRITE2BYTES (ADDRESS, INTENSITY_GREEN, 0x0000); // GREEN at zero intensity
  I2CWRITE2BYTES (ADDRESS, INTENSITY_BLUE, 0x0000); // BLUE at zero intensity
  delay(500); // wait for 0.5s

  // FLash Green for 0.5s
  I2CWRITE2BYTES (ADDRESS, INTENSITY_RED, 0x0000);
  I2CWRITE2BYTES (ADDRESS, INTENSITY_GREEN, 0x0FFF);
  delay(500);

  // Flash Blue for 0.5s
  I2CWRITE2BYTES (ADDRESS, INTENSITY_GREEN, 0x0000); // Blue
  I2CWRITE2BYTES (ADDRESS, INTENSITY_BLUE, 0x0FFF);
  delay(500);
}
```

Figure 20 Test Program Code

The loop routine is an infinite ‘while’ loop that runs for as long as the Arduino board is powered.

After writing the code, save the sketch and click on the "Upload" button in the Arduino editor while the Arduino Uno board is connected to the computer.

The Arduino board and the RGB LED Shield should be connected as shown in Figure 21.

Note: Do not connect a lamp which has a forward voltage of less than 6V or a current rating lower than 300mA.

Power-up the shield and the attached lamp should flash red for 0.5s, then green for 0.5s, and then blue for 0.5s. This routine repeats itself indefinitely.

If no light is seen, check whether the channels are connected to the correct terminals. Ensure that no DMX master is connected to the shield. If a DMX master is connected, turn the DMX512 support off via software with “I2CDMX(DMXOFF)” and re-upload the sketch to the Arduino Uno board.

The colours of the lamp can be changed by simply changing the intensity value parameters using the I2CWRITE2BYTES function.

This sketch does not change any parameters other than the channel intensities, and is therefore probably not optimized for the lamp attached to the shield. The current intensities and off-times used are the default values of the RGB LED Shield. A lamp that has a current rating higher than 300mA and forward voltage higher than 6V will be functional, but the LED current may not be optimized. If these parameters are to be changed, further adjustments to the sketch have to be made.

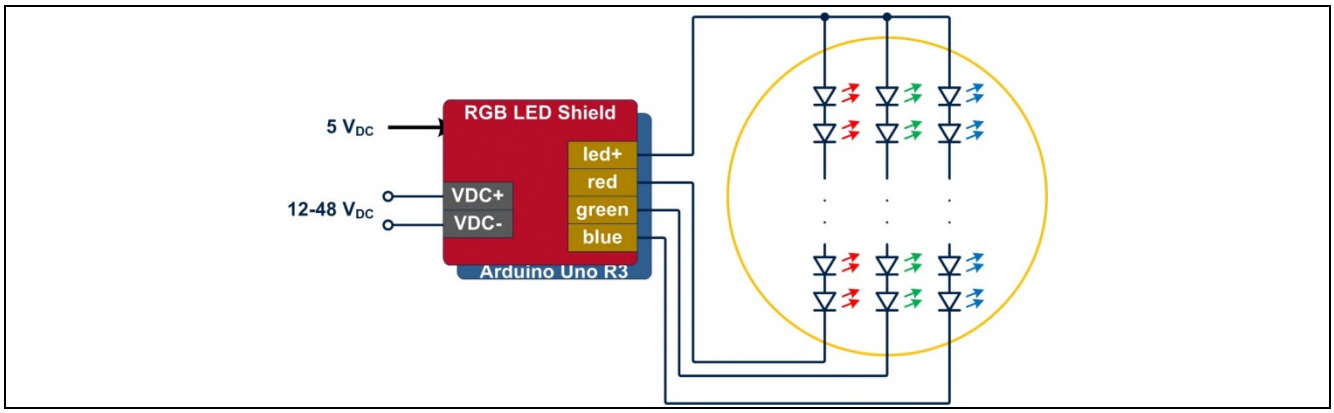


Figure 21 Connecting the master and the Shield

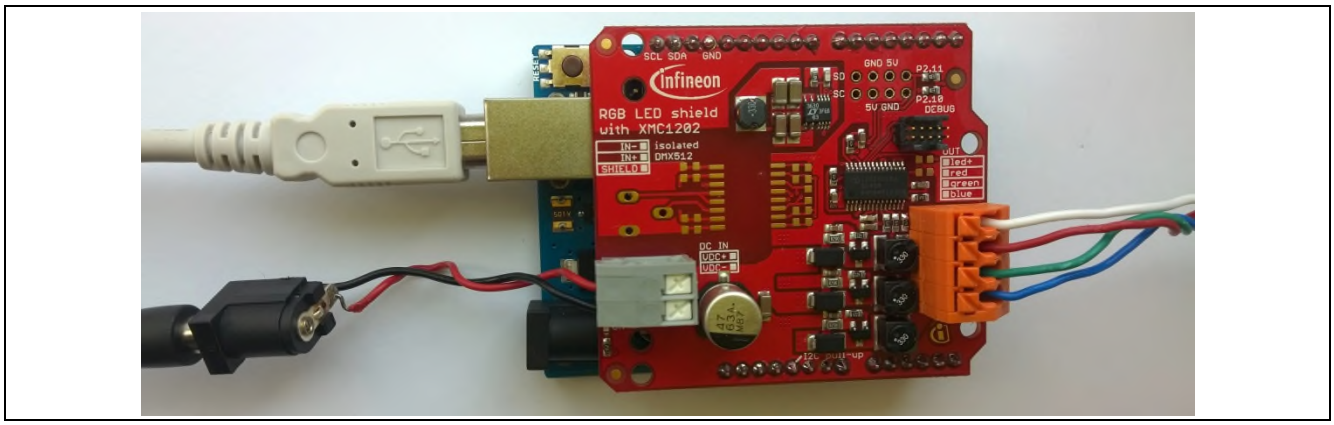


Figure 22 Photo of a connected master and Shield

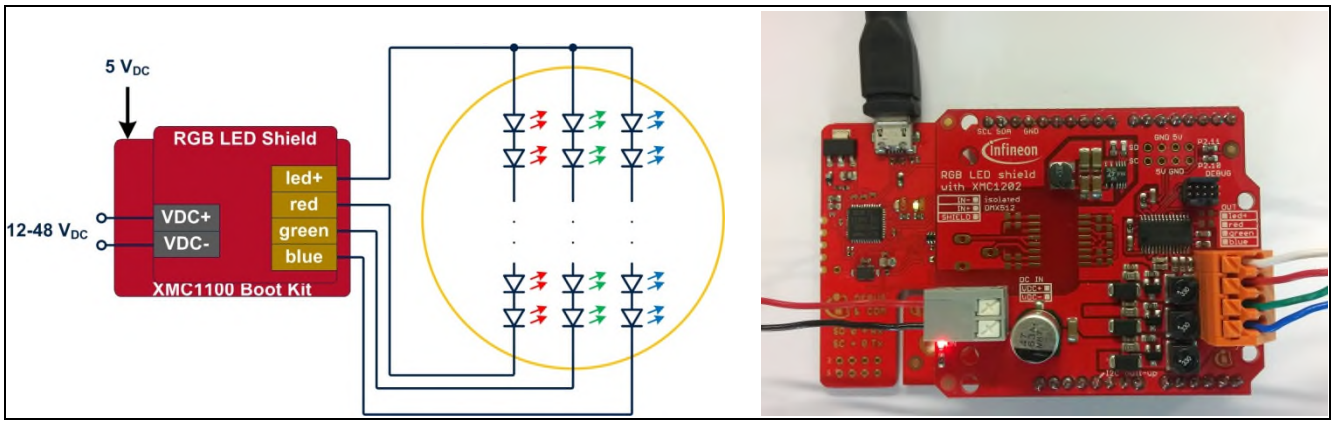


Figure 23 Connecting the Shield to an XMC1100 Boot Kit

5.2 Safe Configuration (DEFAULT)

Every RGB shield is shipped with pre-configured safe parameters. In a safe configuration, the master sends I^2C commands to setup the shield's current intensity and off-time such that almost any attached light engine operates safely at input voltages up to 48V. The important parameters for this basic safe configuration are:

- Peak-current references of red, green and blue channels at 0x15 → around 300mA average current depending on the input voltage and the LED forward voltage
- Off-time of red channel: 0x38 → large ripple with most input voltages and LED forward voltages
- Off-time of green channel: 0x39 → large ripple with most input voltages and LED forward voltages
- Off-time of blue channel: 0x38 → large ripple with most input voltages and LED forward voltages

In the following code snippet, the master board waits for the shield to be powered on before any commands are sent to set up the safe configuration. DMX512 support is turned off, fading is immediate and the dimming level is 0% to ensure that no excessive current flows through the shield and lamp. If the RGB LED Shield responds to the read command, a flag is set (message). If the dimming level in the shield is also at 0% then the shield is considered to be on and ready for new commands.

```
while (on != 1)
{
  I2CDMX (ADDRESS, DMXOFF);
  I2CWRITE2BYTES (ADDRESS, FADERATE, 0x0000);
  I2CWRITE2BYTES (ADDRESS, DIMMINGLEVEL, 0x0000);
  on = I2CREAD(ADDRESS, READ_DIMMINGLEVEL);
  if (message == 1 && on == 0)
  {
    message = 0;
    on = 1;
  }


}
```

Figure 24 Wait for shield to turn on

```
while (redcurr != 0x15 || greencurr != 0x15 || bluecurr != 0x15 || redoff != 0x38 || greenoff != 0x39 || blueoff != 0x38 || brightness != 0)
{
    I2CWRITE6BYTES (ADDRESS, INTENSITY_RGB, 0x0000, 0x000, 0x0000); // Off Light
    // Ensure that parameters are set up correctly. Read back and check. If wrong, write and read again.
    redcurr = I2CREAD (ADDRESS, READ_CURRENT_RED); // Read the red current intensity
    greencurr = I2CREAD (ADDRESS, READ_CURRENT_GREEN); // Read the green current intensity
    bluecurr = I2CREAD (ADDRESS, READ_CURRENT_BLUE); // Read the blue current intensity
    redoff = I2CREAD (ADDRESS, READ_OFFTIME_RED); // Read the off-time of the red channel
    greenoff = I2CREAD (ADDRESS, READ_OFFTIME_GREEN); // Read the off-time of the green channel
    blueoff = I2CREAD (ADDRESS, READ_OFFTIME_BLUE); // Read the off-time of the blue channel
    brightness = I2CREAD (ADDRESS, READ_DIMMINGLEVEL); // Read the dimming level

    I2CWRITE2BYTES (ADDRESS, OFFTIME_RED, 0x38); // Set off-time of red channel to 0x38
    I2CWRITE2BYTES (ADDRESS, OFFTIME_GREEN, 0x39); // Set off-time of green channel to 0x39
    I2CWRITE2BYTES (ADDRESS, OFFTIME_BLUE, 0x38); // Set off-time of blue channel to 0x38
    I2CWRITE2BYTES (ADDRESS, CURRENT_RED, 0x15); // Set current intensity of red channel to 0x15
    I2CWRITE2BYTES (ADDRESS, CURRENT_GREEN, 0x15); // Set current intensity of green channel to 0x15
    I2CWRITE2BYTES (ADDRESS, CURRENT_BLUE, 0x15); // Set current intensity of blue channel to 0x15
    I2CWRITE2BYTES (ADDRESS, DIMMINGLEVEL, 0x0000);
}
```

Figure 25 Code to set Safe Configuration

With the settings in Figure 25, an LED lamp with an output voltage of 6V will still be functional with approximately 300mA average current. The average LED load will be functional with these settings, the peak current will not be exceedingly high, and the ripple not too small to cause problems.

These are also the default values in the RGB LED Shield. If the RGB LED Shield is powered on without an Arduino UNO R3 connected to it, and before any configurations are made with the SAVEPARAMETERS command, these values will be used.

The same block of code can be used to achieve optimized current for the LED lamp. Change the peak-current references and off-times to appropriate values that give an optimized current waveform.

5.3 Configuring the RGB LED Shield

The LED Shield can be programmed to always start up with configured values.

By writing the I2CSAVEPARM (ADDRESS), the master Arduino Board instructs the RGB LED Shield to save the most recent values for off-times, peak-current references, channel intensities, DMX controls, walk time, fade rate and dimming level, to non-volatile memory.

On the next start-up, even without a master Arduino Board, the RGB LED Shield will start with these values loaded into the appropriate registers.

The safe parameters are intended to work with most LED light engines and most input voltages. The drawback is that these parameters are then not optimized. Typically the ripple is too high and the average current is too low.

```
  while (redcurr != 0x15 || greencurr != 0x15 || bluecurr != 0x15 || redoff != 0x38 || greenoff != 0x39 || blueoff != 0x38|| brightness != 0)
  {
      // Ensure that parameters are set up correctly. Read back and check. If wrong, write and read again.
      redcurr = I2CREAD (ADDRESS, READ_CURRENT_RED);
      greencurr = I2CREAD (ADDRESS, READ_CURRENT_GREEN);
      bluecurr = I2CREAD (ADDRESS, READ_CURRENT_BLUE);
      redoff = I2CREAD (ADDRESS, READ_OFFTIME_RED);
      greenoff = I2CREAD (ADDRESS, READ_OFFTIME_GREEN);
      blueoff = I2CREAD (ADDRESS, READ_OFFTIME_BLUE);
      brightness = I2CREAD (ADDRESS, READ_DIMMINGLEVEL);

      I2CWRITE2BYTES (ADDRESS, OFFTIME_RED, 0x38);
      I2CWRITE2BYTES (ADDRESS, OFFTIME_GREEN, 0x39);
      I2CWRITE2BYTES (ADDRESS, OFFTIME_BLUE, 0x38);
      I2CWRITE2BYTES (ADDRESS, CURRENT_RED, 0x15);
      I2CWRITE2BYTES (ADDRESS, CURRENT_GREEN, 0x15);
      I2CWRITE2BYTES (ADDRESS, CURRENT_BLUE, 0x15);
      I2CWRITE2BYTES (ADDRESS, DIMMINGLEVEL, 0x0000);
  }


    I2CWRITE2BYTES (ADDRESS, DMXSLOT, 0x0001); //Set starting slot to 1
    I2CDMX (ADDRESS, DMXON); //On DMX
    I2CWRITE6BYTES (ADDRESS, DMX8BIT, 0x0, 0x1, 0x2); //Configure relevant DMX slots
    I2CWRITE2BYTES (ADDRESS, WALKTIME, 0xF); // Walktime to 150ms
    I2CWRITE2BYTES (ADDRESS, FADERATE, 0x0000); // Immediate dimming
    I2CWRITE2BYTES (ADDRESS, DIMMINGLEVEL, 0xFFF); //Maximum Dimming Level
    I2CWRITE6BYTES (ADDRESS, INTENSITY_RGB, 0x0000, 0x0000, 0x0000);

    I2CSAVEPARAM (ADDRESS); // Save above parameters to Flash. The parameters will be default upon startup of slave
    configStatus = I2CREAD (ADDRESS, READ_CONFIG); // Check if slave has saved those configurations to Flash

  while (configStatus == 0) // If not, save parameters again.
  {
    I2CSAVEPARAM (ADDRESS);
    configStatus = I2CREAD (ADDRESS, READ_CONFIG);
  }

}
```

Figure 26 Configuring the LED Shield

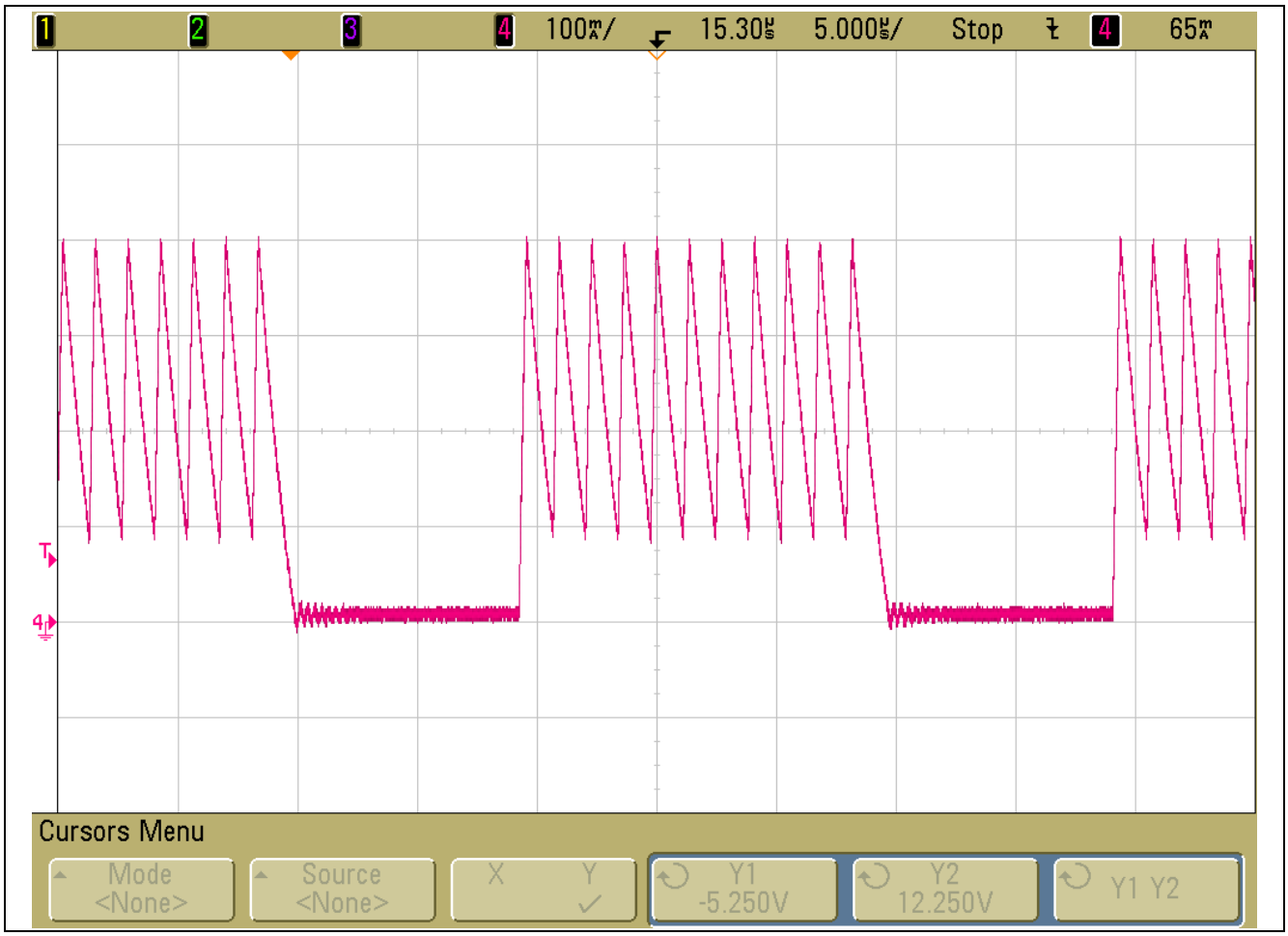


Figure 27 Typical LED current with safe configuration parameters ($48V_{DC}$ input, Traxon Nano Liner XB-9)

5.4 Parameters optimized for Traxon Nano Liner XB-9 with 24V Input Voltage

In this example, the master Arduino board sends I^2C commands to optimize peak-current references and off-times for a Traxon NANO Liner XB-9 linear fixture with high intensity LEDs (**http://www2.traxontechnologies.com/products/product_details/620**).

The NANO Liner XB-9 has forward voltages of 9.1V for the red channel, 8.65V for the green channel and 8.8V for the blue channel.

The important parameters for this configuration are:

- Peak current reference value of Red channel: 0x29
- Peak current reference value of Green channel: 0x30
- Peak current reference value of Blue channel: 0x30
- Off-time of Red channel: 0x18
- Off-time of Green channel: 0x20
- Off-time of Blue channel: 0x20

These parameters result in minimum ripple and 350mA average current, when the Nano Liner XB-9 is used.

Note: Do not use these values for any other LED lamp without checking the current rating and forward voltages.

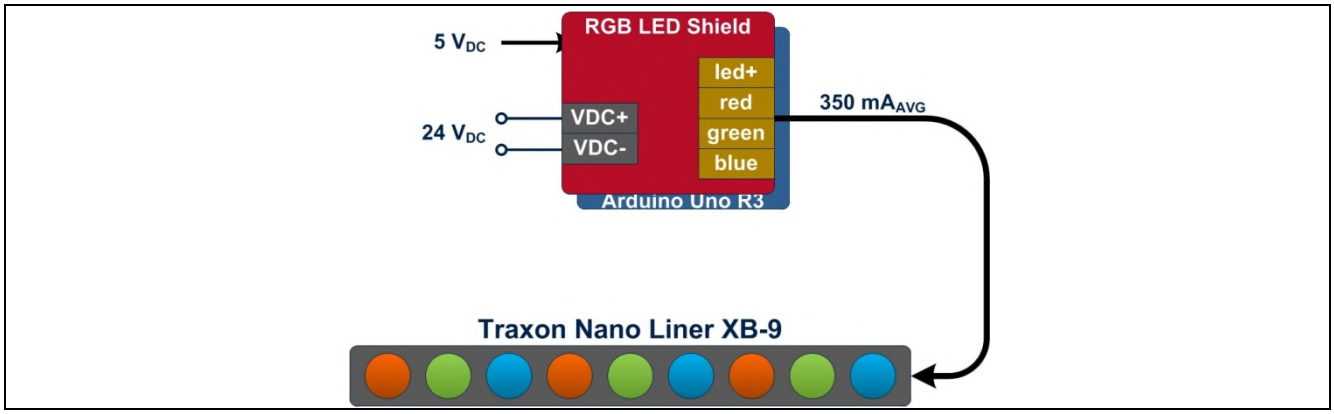


Figure 28 System Block Diagram with Nano Liner XB-9

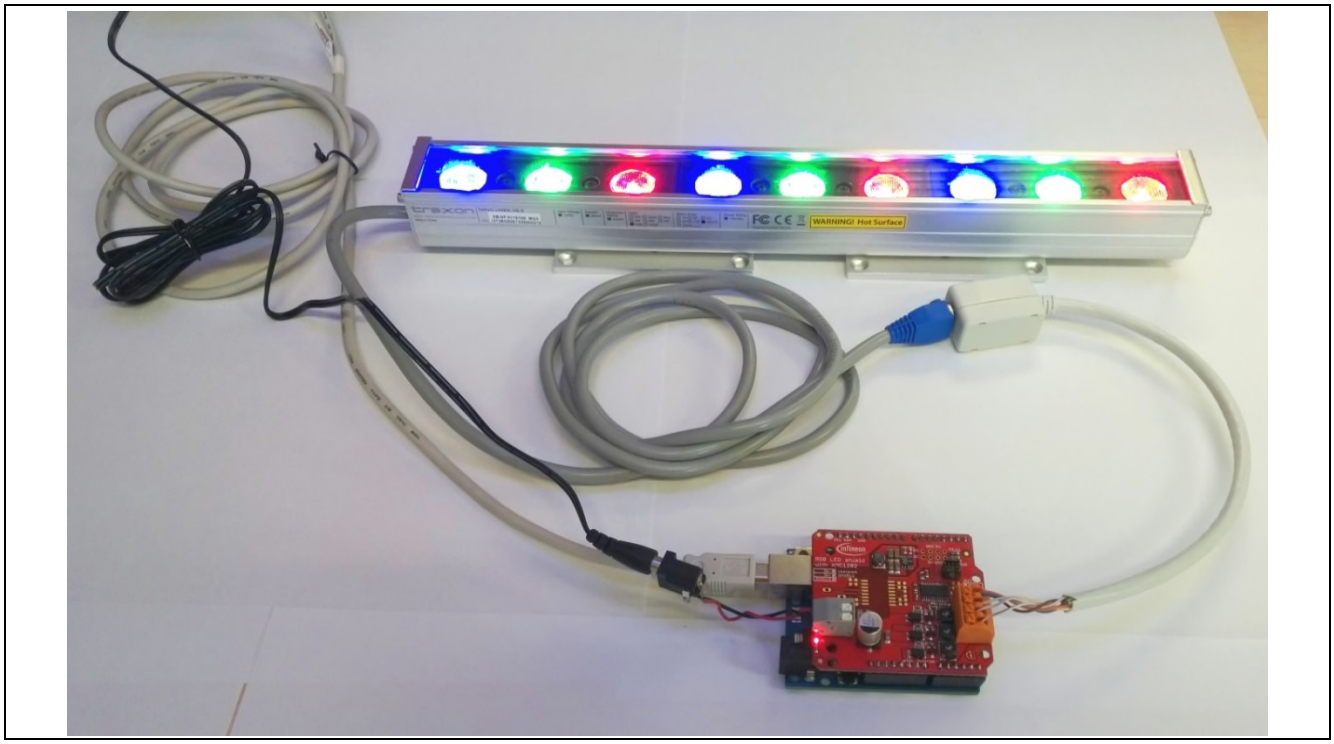


Figure 29 System Photo with Nano Liner XB-9

```
while (redcurr != 0x29 || greencurr != 0x28 || bluecurr != 0x29 || redoff != 0x8 || greenoff != 0x7 || blueoff != 0x7 || brightness != 0)
{
    // Ensure that parameters are set up correctly. Read back and check. If wrong, write and read again.
    redcurr = I2CREAD (ADDRESS, READ_CURRENT_RED);
    greencurr = I2CREAD (ADDRESS, READ_CURRENT_GREEN);
    bluecurr = I2CREAD (ADDRESS, READ_CURRENT_BLUE);
    redoff = I2CREAD (ADDRESS, READ_OFFTIME_RED);
    greenoff = I2CREAD (ADDRESS, READ_OFFTIME_GREEN);
    blueoff = I2CREAD (ADDRESS, READ_OFFTIME_BLUE);
    brightness = I2CREAD (ADDRESS, READ_DIMMINGLEVEL);

    I2CWRITE2BYTES (ADDRESS, OFFTIME_RED, 0x8);
    I2CWRITE2BYTES (ADDRESS, OFFTIME_GREEN, 0x7);
    I2CWRITE2BYTES (ADDRESS, OFFTIME_BLUE, 0x7);
    I2CWRITE2BYTES (ADDRESS, CURRENT_RED, 0x29);
    I2CWRITE2BYTES (ADDRESS, CURRENT_GREEN, 0x28);
    I2CWRITE2BYTES (ADDRESS, CURRENT_BLUE, 0x29);
    I2CWRITE2BYTES (ADDRESS, DIMMINGLEVEL, 0x0000);
    I2CWRITE6BYTES (ADDRESS, INTENSITY_RGB, 0x000, 0x000, 0x000);
}
```

Figure 30 Optimized Parameters for NANO Liner XB-9

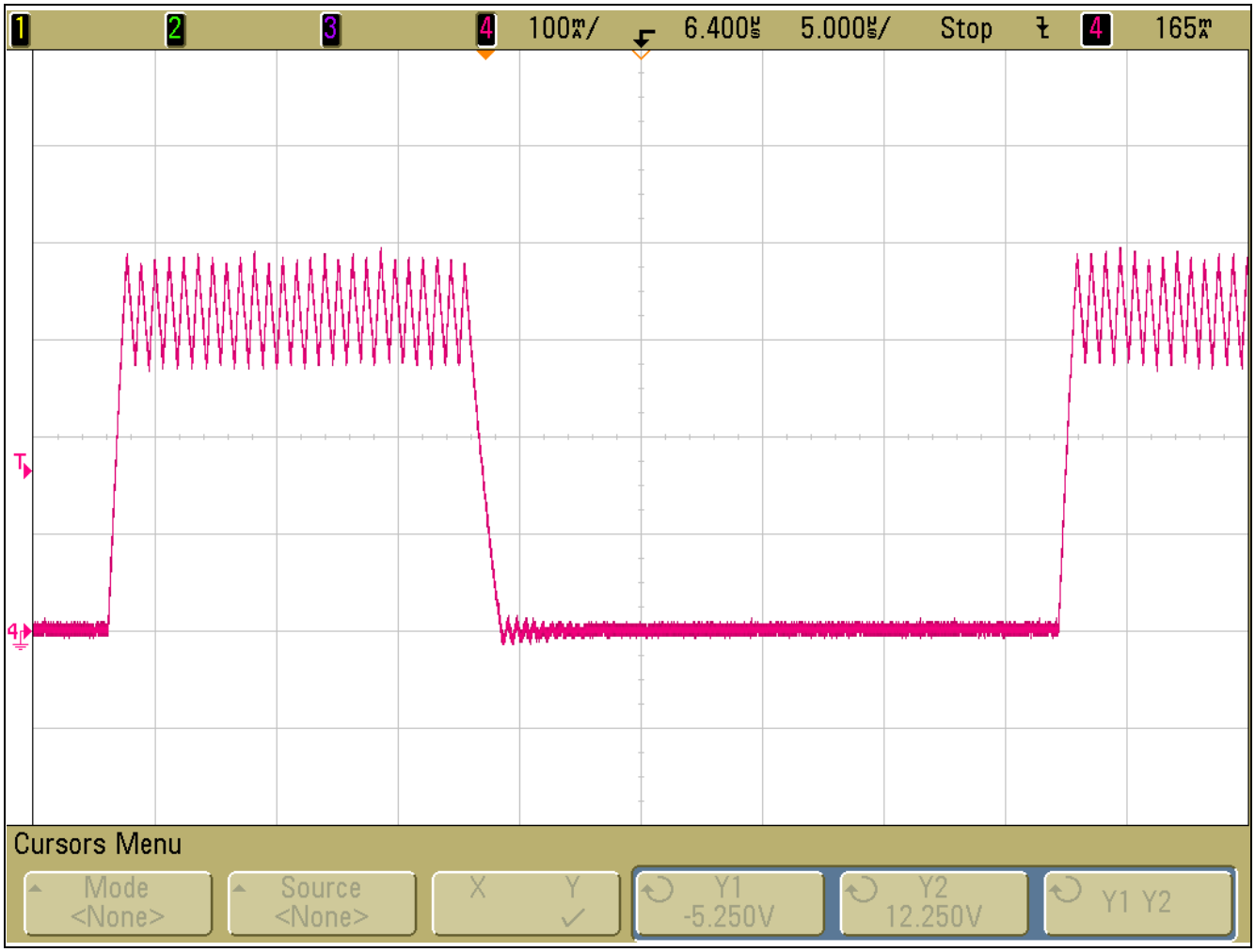


Figure 31 Optimized LED current (red channel, ~40% brightness)

To read values back from the RGB LED shield, send the I2CREAD function. The function itself will print the values to the serial monitor.

To see the values on the serial monitor, initialize the serial port with:

```
Serial.begin(9600)
```

```
Serial.print("Red Int: ");
redint = I2CREAD (ADDRESS, READ_INTENSITY_RED);
Serial.print("Green Int: ");
greenint = I2CREAD (ADDRESS, READ_INTENSITY_GREEN);
Serial.print("Blue Int: ");
blueint = I2CREAD (ADDRESS, READ_INTENSITY_BLUE);
Serial.print("Red Curr: ");
redcurr = I2CREAD (ADDRESS, READ_CURRENT_RED);
Serial.print("Green Curr ");
greencurr = I2CREAD (ADDRESS, READ_CURRENT_GREEN);
Serial.print("Blue Curr: ");
bluecurr = I2CREAD (ADDRESS, READ_CURRENT_BLUE);
Serial.print("Red Off-time: ");
redoff = I2CREAD (ADDRESS, READ_OFFTIME_RED);
Serial.print("Green Off-time: ");
greenoff = I2CREAD (ADDRESS, READ_OFFTIME_GREEN);
Serial.print("Blue Off-time: ");
blueoff = I2CREAD (ADDRESS, READ_OFFTIME_BLUE);
Serial.print("Walk: ");
walk = I2CREAD (ADDRESS, READ_WALKTIME);
Serial.print("Brightness: ");
brightness = I2CREAD (ADDRESS, READ_DIMMINGLEVEL);
Serial.print("FadeTime: ");
fadetime = I2CREAD (ADDRESS, READ_FADERATE);
Serial.print("DMX Red H: ");
dmx_RedH = I2CREAD (ADDRESS, READ_DMXREDH);
Serial.print("DMX Red L: ");
dmx_RedL = I2CREAD (ADDRESS, READ_DMXREDL);
Serial.print("DMX GREEN H: ");
dmx_GreenH = I2CREAD (ADDRESS, READ_DMXGREENH);
Serial.print("DMX GREEN L: ");
dmx_GreenL = I2CREAD (ADDRESS, READ_DMXGREENL);
Serial.print("DMX BLUE H: ");
dmx_BlueH = I2CREAD (ADDRESS, READ_DMXBLUEH);
Serial.print("DMX BLUE L: ");
dmx_BlueL = I2CREAD (ADDRESS, READ_DMXBLUEL);
Serial.print("DMX Enable: ");
dmx_enable = I2CREAD (ADDRESS, READ_DMX);
Serial.print("DMX Starting Slot: ");
dmx_starting = I2CREAD (ADDRESS, READ_DMXSLOT);
```

Figure 32 Reading values from the RGB LED shield

5.5 Parameters optimized for Traxon Nano Liner XB-18 with 48V Input Voltage

http://www2.traxontechnologies.com/products/product_details/620

The NANO Liner XB-18 has forward voltages of 16.3V for the red channel, 17.6V for the green channel and 17.6V for the blue channel.

The important parameters for this configuration are:

- Peak current reference value of Red channel: 0x27
- Peak current reference value of Green channel: 0x29
- Peak current reference value of Blue channel: 0x29
- Off-time of Red channel: 0x8
- Off-time of Green channel: 0x8
- Off-time of Blue channel: 0x7

These parameters result in minimum ripple at 350mA average current when the Nano Liner XB-18 is used.

Note: Do not use these values for any other LED lamp without checking the current rating and forward voltages.

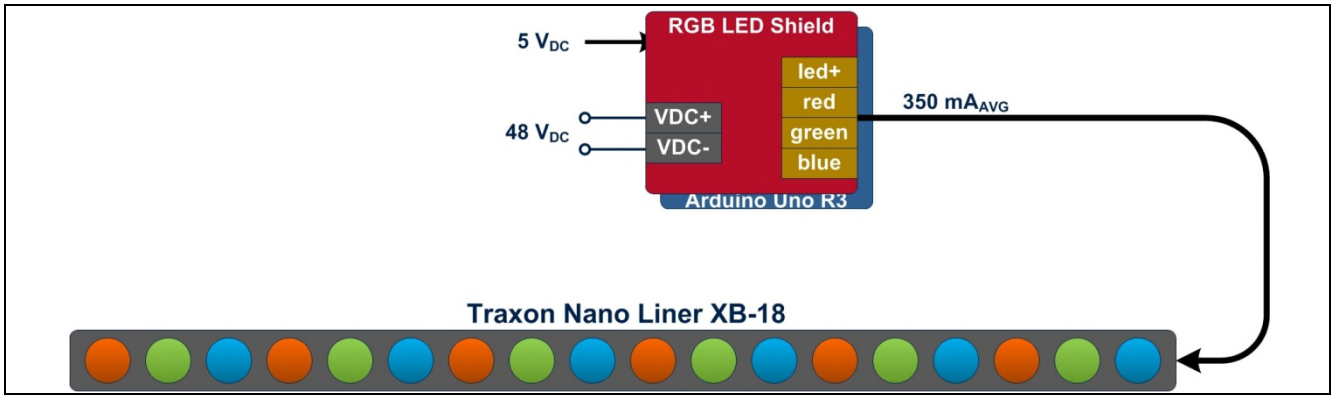


Figure 33 System Block Diagram with Nano Liner XB-18

```
while (redcurr != 0x29 || greencurr != 0x28 || bluecurr != 0x29 || redoff != 0x8 || greenoff != 0x7 || blueoff != 0x7 || brightness != 0)
{
    // Ensure that parameters are set up correctly. Read back and check. If wrong, write and read again.
    redcurr = I2CREAD (ADDRESS, READ_CURRENT_RED);
    greencurr = I2CREAD (ADDRESS, READ_CURRENT_GREEN);
    bluecurr = I2CREAD (ADDRESS, READ_CURRENT_BLUE);
    redoff = I2CREAD (ADDRESS, READ_OFFTIME_RED);
    greenoff = I2CREAD (ADDRESS, READ_OFFTIME_GREEN);
    blueoff = I2CREAD (ADDRESS, READ_OFFTIME_BLUE);
    brightness = I2CREAD (ADDRESS, READ_DIMMINGLEVEL);

    I2CWRITE2BYTES (ADDRESS, OFFTIME_RED, 0x8);
    I2CWRITE2BYTES (ADDRESS, OFFTIME_GREEN, 0x7);
    I2CWRITE2BYTES (ADDRESS, OFFTIME_BLUE, 0x7);
    I2CWRITE2BYTES (ADDRESS, CURRENT_RED, 0x29);
    I2CWRITE2BYTES (ADDRESS, CURRENT_GREEN, 0x28);
    I2CWRITE2BYTES (ADDRESS, CURRENT_BLUE, 0x29);
    I2CWRITE2BYTES (ADDRESS, DIMMINGLEVEL, 0x0000);
    I2CWRITE6BYTES (ADDRESS, INTENSITY_RGB, 0x000, 0x000, 0x000);
}
```

Figure 34 Optimized Parameters for NANO Liner XB-18

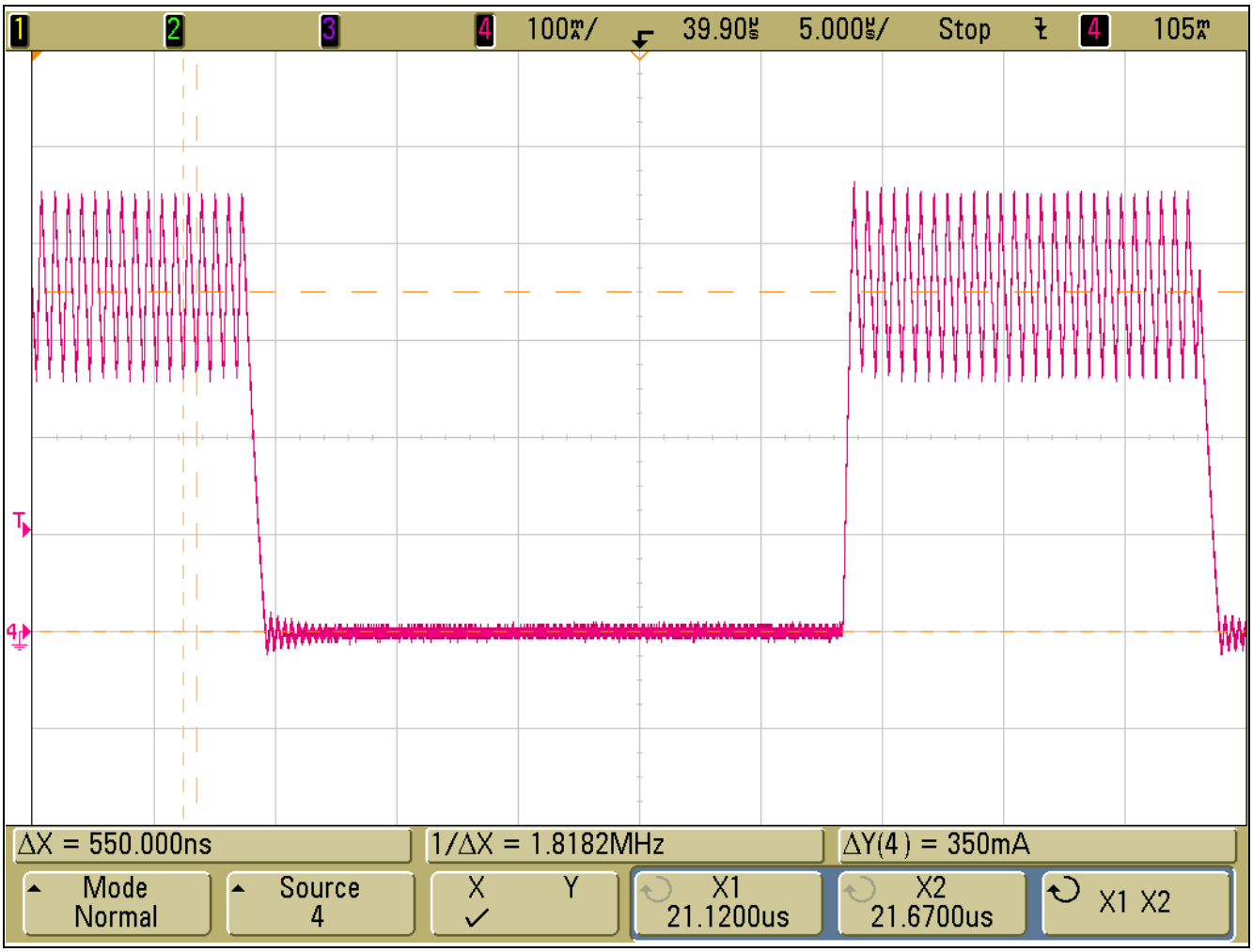


Figure 35 Optimized LED current (red channel, ~35% brightness)

Setting the Parameters for YOUR LED Lamp

6 Parameter Setup for YOUR LED Lamp

For small ripple and accurate average current, the two main parameters have to be carefully set up for every LED channel. By default these parameters are safe values that, while functional with most light engines and input voltages, produce a large ripple and typically an average current that is too low.

To ensure the correct values are used:

1. Decide what LED light engine will be used.
   - Read the manual carefully.
   - Check the forward voltage of every LED channel. Any value between 6V and 48V should be fine.
   - Check the optimum forward current of every LED channel. The average value can be maximum 700mA and the peak can be maximum 1A.
2. Decide what DC input voltage will be used.
   - It has to be higher than the highest forward voltage of the LED channels.
   - Any value between 12V and 48V should be fine but values higher than four times the LED forward voltages should be avoided if small ripple is desired.
   - Generally, the higher the ratio between the input voltage and the forward voltage, the more difficult it is to set up the peak-current reference and the off-time.
3. If the light engine allows an average current of at least 300mA, try using the safe parameters.
4. Get an oscilloscope with a current probe and measure the current of one LED string at a time.
5. Roughly set up the peak current by increasing the peak-current reference in small steps.
6. Decrease the ripple by decreasing the off-time in small steps.
7. Repeat steps 5 and 6 until the ripple is acceptable and the average current is accurate.
8. Repeat steps 5, 6 and 7 until all channels are configured.
9. Enjoy the light.

Table 9 Parameters optimized for different light engines, LED currents, and input voltages

Input Voltage	Light Engine	Average LED Current	Channel	Forward Voltage	Peak-Current Reference Parameter	Peak-Current Reference*	Off-Time Parameter	Generated Off-Time**
12-48 V_{DC}	Safe parameters	~300 mA, depending on the input voltage	Red	? V	*0x15*	128 mA	*0x38*	875 ns
			Green	? V	*0x15*	128 mA	*0x39*	891 ns
			Blue	? V	*0x15*	128 mA	*0x38*	875 ns
24 V_{DC}	LedEngin LZC-83MC00	500 mA	Red	9.4 V	*0x46*	427 mA	*0x07*	109 ns
		550 mA	Green	16.8 V	*0x51*	494 mA	*0x05*	78 ns
		700 mA	Blue	14.0 V	*0x72*	696 mA	*0x04*	63 ns
48 V_{DC}	LedEngin LZC-83MC00	700 mA	Red	9.4 V	*0x35*	323 mA	*0x19*	391 ns
			Green	16.8 V	*0x43*	409 mA	*0x12*	281 ns
			Blue	14.0 V	*0x61*	592 mA	*0x10*	250 ns
24 V_{DC}	Traxon Nano Liner XB-9	350 mA	Red	9.1 V	*0x29*	250 mA	*0x08*	125 ns
			Green	8.7 V	*0x28*	244 mA	*0x07*	109 ns

Input Voltage	Light Engine	Average LED Current	Channel	Forward Voltage	Peak-Current Reference Parameter	Peak-Current Reference*	Off-Time Parameter	Generated Off-Time**
			Blue	8.8 V	*0x29*	250 mA	*0x07*	109 ns
24 V_{DC}	Traxon Nano Liner XB-18	350 mA	Red	16.3 V	*0x39*	348 mA	*0x03*	47 ns
			Green	17.6 V	*0x39*	348 mA	*0x03*	47 ns
			Blue	17.6 V	*0x39*	348 mA	*0x03*	47 ns
48 V_{DC}	Traxon Nano Liner XB-18	350 mA	Red	16.3 V	*0x27*	238 mA	*0x08*	125 ns
			Green	17.6 V	*0x29*	250 mA	*0x08*	125 ns
			Blue	17.6 V	*0x29*	250 mA	*0x07*	109 ns

* The LED current always overshoots the peak-current reference so the actual peak current is higher than the peak-current reference.

** The actual off-time is longer than the generated off-time due to propagation delay and the fact that the fixed off-time starts to get generated only after the LED current falls below the peak-current reference.

APPENDIX

7 Appendix

7.1 Description of the I²C Functions Provided

The functions provided for use with the Arduino Uno R3 board and XMC1100 Boot Kit are described.

7.1.1 I2CWRITE2BYTES (ADDRESS, COMMAND, DATA)

This function is used when only one Word of data is to be sent.
The function parameters are:

- The RGB LED Shield's address
- A command
- An unsigned 16-bit integer, sent as 2 bytes

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. 8-bit command
5. Upper 8-bit of word
6. Lower 8-bit of word
7. STOP condition

7.1.2 I2CWRITE6BYTES (ADDRESS, COMMAND, DATA, DATA, DATA)

This function is used when three Words of data are to be sent.
The function parameters are:

- The RGB LED Shield's address
- A command
- An unsigned 16-bit integer sent as 2 bytes

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. 8-bit command
5. Upper 8-bit of first data
6. Lower 8-bit of first data
7. Upper 8-bit of second word
8. Lower 8-bit of second word
9. Upper 8-bit of third word
10. Lower 8-bit of third word
11. STOP condition

7.1.3 I2CWRITE12BYTES (ADDRESS, COMMAND, DATA, DATA, DATA, DATA, DATA, DATA)

This function is used when six Words of data are to be sent.

The function parameters are:

- The RGB LED Shield's address
- A command
- An unsigned 16-bit integer sent as 2 bytes

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. 8-bit command
5. Upper 8-bit of first word
6. Lower 8-bit of first word
7. Upper 8-bit of second word
8. Lower 8-bit of second word
9. Upper 8-bit of third word
10. Lower 8-bit of third word
11. Upper 8-bit of fourth word
12. Lower 8-bit of fourth word
13. Upper 8-bit of fifth word
14. Lower 8-bit of fifth word
15. Upper 8-bit of sixth word
16. Lower 8-bit of sixth word
17. STOP condition

7.1.4 I2CREAD (ADDRESS, COMMAND)

This function is used to read a parameter value. 16-bits of data will be received as 2 bytes and returned by the function.

The function parameters are:

- The RGB LED Shield's address
- A command

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. 8-bit command
5. Repeated START condition
6. 1st Address byte with 'Zero' for a transmission request
7. 2nd Address byte
8. 1st Address byte with 'One' for a request for data
9. Acknowledge
10. Not-acknowledge
11. STOP condition

7.1.5 I2CREAD_DIRECTACCESS (ADDRESS, REGISTER ADDRESS)

This function is used when a value in a specific register is to be read. With this function it is possible to access any value contained in the RGB LED Shield registers which may not already be accessible by the provided commands.

The function parameters are:

- The RGB LED Shield's address
- The address of the target register
  - The register address is a 32-bit value and will be sent in 4 bytes.

32-bits of data will be received in 4 bytes and returned by the function.

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. 8-bit command: READ_DIRECTACCESS
5. Bits 0 – 7 of register address
6. Bits 8 – 15 of register address
7. Bits 16 – 23 of register address
8. Bits 24 – 31 of register address
9. Repeated START condition
10. 1st Address byte with 'Zero' for a transmission request
11. 2nd Address byte
12. 1st Address byte with 'One' for a request for data
13. Acknowledge

14.Acknowledge
15.Acknowledge
16.Not-acknowledge
17.STOP condition

7.1.6 I2CWRITE_DIRECTACCESS (ADDRESS, COMMAND, REGISTER ADDRESS, DATA)

This function should be used when a value in a specific register is to be read.
Any of the RGB LED Shield registers can be accessed with this function.
The function parameters are:

- The RGB LED Shield's address
- A command
- The address of the target register

Both the register address and data are 32-bit values and will each be sent in 4 bytes.

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. 8-bit command
5. Bits 0 – 7 of register address
6. Bits 8 – 15 of register address
7. Bits 16 – 23 of register address
8. Bits 24 – 31 of register address
9. Repeated START condition
10. 1st Address byte with 'Zero' for a transmission request
11. 2nd Address byte
12. 'One' for a request for data
13. Bits 0 – 7 of data
14. Bits 8 – 15 of data
15. Bits 16 – 23 of word
16. Bits 24 – 31 of word
17. STOP condition

7.1.7 I2CCHANGEADDRESS (ADDRESS, NEW ADDRESS)

This function is used to change the 10-bit address of the RGB LED Shield.

Using this function by itself will only result in a temporary change in address. On the next startup the RGB LED Shield's address will revert to the default 0x15E.

To permanently change the address, this command must be used in conjunction with the I2CSAVEPARAM function.

The function parameters are:

- The RGB LED Shield's address
- The new address
  - The new address is a 10-bit value, but the function will send it as 111100XX, where XX are the 2 most significant bits of the new address as one byte, and the lower 8-bits as a second byte. This format follows the I^2C specification for 10-bit addressing.

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. CHANGEADDRESS command
5. 111100XX, where XX is the 2 most significant bits of the new address
6. Lower 8-bits of address
7. STOP condition

7.1.8 I2CDMX (ADDRESS, DMXCOMMAND)

This function enables or disables the DMX control.

The function parameters are:

- The RGB LED Shield's address
- A DMX Control related command

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. 8-bit command
5. STOP condition

7.1.9 I2CSAVEPARAM (ADDRESS)

This function saves the current parameters in the shield to its Flash memory. The function will send a read request for the shield to respond when the Flash access operation is finished.

The function parameter is the RGB LED Shield's address.

The data transfer sequence is:

1. START condition
2. 1st Address byte with 'Zero' for a transmission request
3. 2nd Address byte
4. SAVEPARAMETERS command
5. Repeated START condition
6. 1st Address byte with 'Zero' for a transmission request
7. 2nd Address byte
8. 1st Address byte with 'One' for a request for data
9. Acknowledge
10. Not-acknowledge
11. STOP condition

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG

LifeElectronics
Живое партнерство

ООО «ЛайфЭлектроникс» **“LifeElectronics” LLC**

ИНН 7805602321 КПП 780501001 Р/С 40702810122510004610 ФАКБ "АБСОЛЮТ БАНК" (ЗАО) в г.Санкт-Петербурге К/С 30101810900000000703 БИК 044030703

Компания «Life Electronics» занимается поставками электронных компонентов импортного и отечественного производства от производителей и со складов крупных дистрибьюторов Европы, Америки и Азии.

С конца 2013 года компания активно расширяет линейку поставок компонентов по направлению коаксиальный кабель, кварцевые генераторы и конденсаторы (керамические, пленочные, электролитические), за счёт заключения дистрибьюторских договоров

Мы предлагаем:

- *Конкурентоспособные цены и скидки постоянным клиентам.*
- *Специальные условия для постоянных клиентов.*
- *Подбор аналогов.*
- *Поставку компонентов в любых объемах, удовлетворяющих вашим потребностям.*

- *Приемлемые сроки поставки, возможна ускоренная поставка.*
- *Доставку товара в любую точку России и стран СНГ.*
- *Комплексную поставку.*
- *Работу по проектам и поставку образцов.*
- *Формирование склада под заказчика.*

- *Сертификаты соответствия на поставляемую продукцию (по желанию клиента).*
- *Тестирование поставляемой продукции.*
- *Поставку компонентов, требующих военную и космическую приемку.*
- *Входной контроль качества.*
- *Наличие сертификата ISO.*

В составе нашей компании организован Конструкторский отдел, призванный помогать разработчикам, и инженерам.

Конструкторский отдел помогает осуществить:

- *Регистрацию проекта у производителя компонентов.*
- *Техническую поддержку проекта.*
- *Защиту от снятия компонента с производства.*
- *Оценку стоимости проекта по компонентам.*
- *Изготовление тестовой платы монтаж и пусконаладочные работы.*

Тел: +7 (812) 336 43 04 (многоканальный)
Email: org@lifeelectronics.ru

www.lifeelectronics.ru