

## **Introduction**

This demo board features six multi-channel RGB linear drivers: AL5887/AL5887Q (36 channels), AL58818/AL58818Q (18 channels), and AL58812/AL58812Q (12 channels). The main goal is to show vivid LED effects by communicating through a digital interface (I2C or SPI). This demo board describes a built-in MCU with pre-loaded programming code to light up three separate rings of RGB LEDs independently, with the ability to select jumpers. Users may also hook up to their own LEDs through external connectors and the external interface connector to perform their own programming (refer to page 3 - Figure 3).

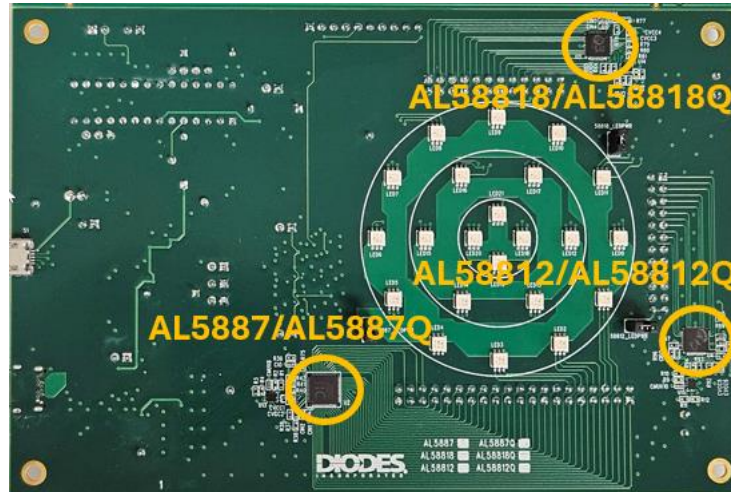


Figure 1: Top View of the Demo Board















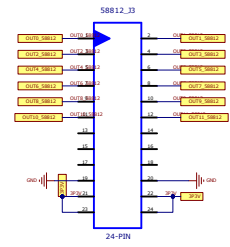
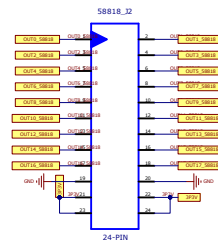




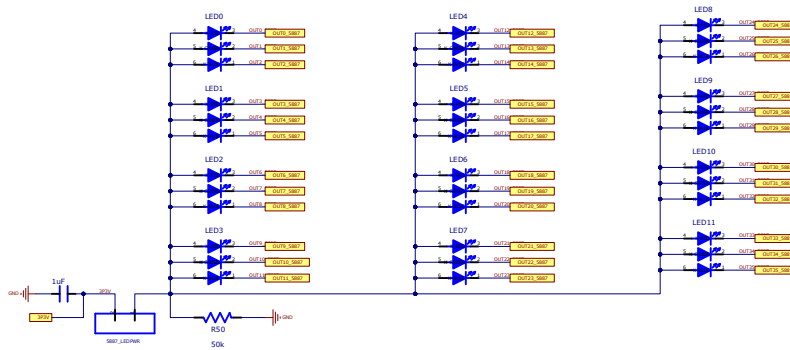
### Schematic (continued)

Headers for external LED Display Driving

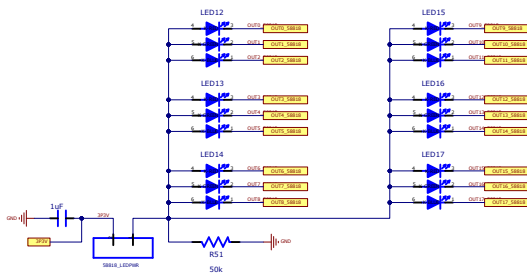
NOTE TO LAYOUT DESIGNER, Use Net names on each header pin as silkscreen to the PCB



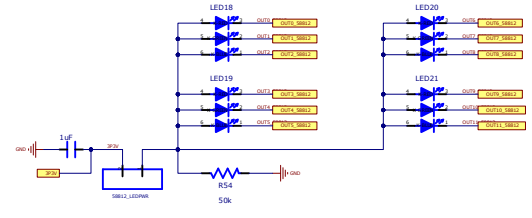
AL5887 LEDs. Place 12 LEDs in a ring. This is the outer most ring



AL58818 LEDs. Place 6 LEDs in a ring. This is the middle ring



AL58812 LEDs. Place 4 LEDs in a ring. This is the inner most ring



**Bill of Materials**

Location	Description	MFR	P/N	Package
CBUCKVIN1, CAREF, CAVCC, CAVCC1-CAVCC3, CDTR, CIN2, CIN3, CIN5, CIO, CIO1, CIO2, CMUX1-CMUX10, CUSB_B1, CVCC1, CVCC3, CVCC5	CAP CER 0.1UF 6.3V X5R	KYOCERA AVX	04026D104KAT4A	402
CBUCKVIN2-4 CBUCKVIN25 CVCC2 CUSB_B2 CVCC4 CUCAP CVCC6	CAP CER 1UF 6.3V X5R	Taiyo Yuden	JMK105BJ105KV-F	402
CBUCKVOUT1-4	CAP CER 22UF 6.3V X5R	Samsung Electro-Mechanics	CL21A226MQQNNNE	805
CBUCKVOUT5-6	CAP CER 10UF 6.3V X5R	Samsung Electro-Mechanics	CL21A106KQFNNE	805
CIN1 CIN6 CIN4	CAP CER 10UF 6.3V X5R	Samsung Electro-Mechanics	CL05A106MQ5NUNC	402
CXTAL1-4	CAP CER 22PF 50V C0G/NP0	KYOCERA AVX	KGM05ACG1H220FH	402
CVBUS1-2	CAP TANT 47UF 10% 10V	KYOCERA AVX	TAJB476K010RNJ	1411 (3528 metric)
G1-2	CRYSTAL 16.0000MHZ 20PF	ECS Inc.	ECS-160-20-3X-TR	SMD
D1-2 D5-6	DIODE, GENERAL, 1N914, 100V, DO-35	DIODES	1N4448-WS-7	-
F1	Fuses with Leads - Through Hole 125V 1A MICRO Very Fast Acting	LITTLEFUSE	0273001.H	-
L1	Ferrite Bead, SMT, 60 Ohm	Murata	BLM21PG600SH1D	-
L2	INDUCTOR, 10 uH, PMI Power Multilayer Inductor	WURTH ELEKTRONIK	74479887310A	-
L3	INDUCTOR, SMT, 1.2uH	COILCRAFT	XGL5020-122MEC	-
LED22 LEDRX LEDTX USBC_5V	LED, SMD GREEN DIFFUSED	Lumex Opto/Components Inc.	SML-LX1206GW-TR	1206
Q1	MOSFET N-CHANNEL 60V 150mA	DIODES	2N7002Q	SOT-23
U1	8-Bit Microcontroller	ATMEL	ATMEGA16U2-MU	SMD-32
U4	<b>AL58812/AL58812Q, I2C/SPI 12-CHANNEL RGB LED DRIVER</b>	<b>DIODES</b>	<b>AL58812/AL58812Q</b>	<b>QFN5050-40</b>
U3	<b>AL58818/AL58818Q, I2C/SPI 18-CHANNEL RGB LED DRIVER</b>	<b>DIODES</b>	<b>AL58818/AL58818Q</b>	<b>QFN5050-40</b>
U2	<b>AL5887/AL5887Q, I2C/SPI 36-CHANNEL RGB LED DRIVER</b>	<b>DIODES</b>	<b>AL5887/AL5887Q</b>	<b>QFN6060-52</b>
U6-15	<b>Low Voltage Dual SPDT Analog Switch</b>	<b>DIODES</b>	<b>PI5A23157ZUAEX</b>	<b>UQFN-10</b>
U5	Low-Power CMOS 8-bit Microcontroller w/ Arduino Bootloader	ATMEL	ATMEGA328P-PU Arduino Bootloader	THD-28
U16	<b>SYNCHRONOUS BUCK CONVERTER</b>	<b>DIODES</b>	<b>AP62800</b>	<b>SMD-12</b>
LED0-21	SMD Tricolor White Surface LED	ANY	ASMT-YTD2-0BB02	SMD-6

**Bill of Materials (continued)**

Location	Description	MFR	P/N	Package
S1	MICO USB TYPE B RECEPTACLE	Amphenol FCi	10118194	USB-5LD
RST_5812 RST_5818 RST_5887 RST_16U1-4	SWITCH, PUSH-BUTTON	TE	FSM2JSMA	SMD-4PIN
S2	USB3.2 Gen2 TYPE C RECEPTACLE	WWW.GCT.CO	USB4056	USB-24LD
PGND1-5	TESTPOINT, CLIP, BLACK .075"DIA	KEYSTONE	5001	THD-1PIN
PC0-3 PD2-3 PD5-7 VCC_5887 VCC_588_12 VCC_588_18	TESTPOINT, CLIP, RED .075"DIA	Lumex Opto/Components Inc.	SML-LX1206GW-TR	THD-1PIN
_3P3V_5VUSBB _5VUSBC	TESTPOINT, CLIP-LEAD	KEYSTONE	5016	SMT
R16-R22, R31-R36, R39, R43-R46, R52, R61, R65, R78, R84, R91, R96, R100, R102, R106-R107, R23, R25, R70, R71, R74, R82, R98, R99	RES SMD 0 OHM JUMPER 1/10W	Panasonic Electronic Components	ERJ-2GE0R00X	0402
R27	RES SMD 1K OHM 0.1% 1/8W	YAGEO	RT0805BRD071KL	0805
R29 R40 R49 R79 R92	RES SMD 100K OHM 0.1% 1/16W	YAGEO	RT0402BRD07100KL	0402
R55, R56, R75, R76, R67, R85, R89, R97, R69, R37	RES SMD 10K OHM 0.1% 1/16W	YAGEO	RT0402BRE0710KL	0402
R48	RES 137K OHM 0.1% 1/10W	TE Connectivity Passive Product	RP73PF1E137KBD	0402
R104, R105, R108	RES SMD 1K OHM 0.1% 1/16W	YAGEO	RT0402BRE071KL	0402
R66 R68	RES 1M OHM 1% 1/16W	Walsin Technology Corporation	WR04X1004FTL	0402
R41 R80 R93	RES SMD 2.1K OHM 0.1% 1/16W	Susumu	RG1005P-2101-B-T5	0402
R62-63	RES 22 OHM 5% 1/16W	Walsin Technology Corporation	WR04X220 JTL	0402
R47	RES SMD 30.1KOHM 0.1% 1/16W	YAGEO	RT0402BRE0730K1L	0402
R1 R3 R5 R8 R10-11 R13-15 R30 R88	RES SMD 300 OHM 0.1% 1/16W	YAGEO	RT0402BRD07300RL	0402
R59 R101 R103	RES SMD 432 OHM 0.1% 1/16W	YAGEO	RT0402BRD07432RL	0402
R60	RES 499K OHM 0.1% 1/16W	YAGEO	RC0402BR-07499KL	0402
R28	RES SMD 48.7KOHM 0.1% 1/16W	YAGEO	RT0402BRE0748K7L	0402
R57-58	RES SMD 5.1K OHM 0.1% 1/16W	YAGEO	RT0402BRD075K1L	0402
R50-51 R54	RES SMD 49.9KOHM 0.1% 1/16W	YAGEO	RT0402BRE0749K9L	0402
J1	Headers & Wire Housings Classic PCB Header Strips, 0.100" pitch	SAMTEC	TSW-110-07-T-S	THD-10
58812_J3 58818_J2	CONN., HEADER, 12X2-PIN, 100cc	SAMTEC	TSW-112-07-S-D	THD-24
5887_J1	CONN., HEADER, 40- PIN 2-ROW, .100cc	SAMTEC	TSW-120-07-T-D	THD-40

**Bill of Materials (continued)**

Location	Description	MFR	P/N	Package
16U2_1-4 328P_1-3 3P3BUCK_EN 58812_LEDPWR 58818_LEDPWR 5887_LEDPWR EXT/INT_SELECT SPI/I2C_SELECT	CONN., HEADER, 2-PIN 1-ROW, .100cc	SAMTEC	TSW-102-07-T-S	THD-2
<b>D3, D4, R24, R26, R72, R73, R83, R86, R87, R95, R2, R4, R6, R7, R9, R12, R38, R42, R53, R64, R77, R81, R90, R94</b>	<b>Do Not Install components</b>	-	-	-

**Example “C” Code Pre-Loaded in the MCU to Light Up All 3 RGB LED Rings**

```

/* Author: Diodes INC */
/* Date: 3/1/2024 */
/* Company: Diodes Incorporated */
/*****/

#include<Wire.h>
#define I2C_Addr 0x30 // I2C address
#define I2C_Addr0 0x30 // I2C address
#define I2C_Addr1 0x31 // I2C address
#define I2C_Addr2 0x32 // I2C address
#define I2C_Addr3 0x33 // I2C address
#define I2C_AddrU 0x1c // I2C address

// main function that allows us to communicate with the chip
void writeByte(uint8_t deviceAddress, uint8_t registerAddress, uint8_t registerData) {
  Wire.beginTransmission(deviceAddress); // sends device address and starts communication
  Wire.write(registerAddress); // sends register address
  Wire.write(registerData); // sends register data
  Wire.endTransmission(); // stops communication
}

// put your setup code here, to run once:
void setup() {
  Wire.begin();
  Wire.setClock(200000); // set I2C to run at 200kHz
  initialize();
}

// put your main code here, to run repeatedly:
void loop() {
  mode1(); // change this to whatever mode is desired
}

void initialize() {
  // setup the board ADDRESS30
  writeByte(I2C_Addr0, 0x00, 0x40); // write a 1 to CHIP_EN
  writeByte(I2C_Addr0, 0x38, 0xFF); // write a 1 to CHIP_EN
  writeByte(I2C_Addr0, 0x00, 0x40); // write a 1 to CHIP_EN
  for (uint8_t i = 0x08; i <= 0x13; i++) { // start at first brightness register and go to the last
    writeByte(I2C_Addr0, i, 0x80); // write all brightness to half
  }
  for (uint8_t i = 0x14; i <= 0x37; i++) { // start at first color register and go to the last
    writeByte(I2C_Addr0, i, 0x00); // write all color to 0
  }
  // setup the board ADDRESS31
  writeByte(I2C_Addr1, 0x00, 0x40); // write a 1 to CHIP_EN
  writeByte(I2C_Addr1, 0x38, 0xFF); // write a 1 to CHIP_EN
  writeByte(I2C_Addr1, 0x00, 0x40); // write a 1 to CHIP_EN
  for (uint8_t i = 0x08; i <= 0x13; i++) { // start at first brightness register and go to the last
    writeByte(I2C_Addr1, i, 0x80); // write all brightness to half
  }
  for (uint8_t i = 0x14; i <= 0x37; i++) { // start at first color register and go to the last
    writeByte(I2C_Addr1, i, 0x00); // write all color to 0
  }
  // setup the board ADDRESS32
  writeByte(I2C_Addr2, 0x00, 0x40); // write a 1 to CHIP_EN
  writeByte(I2C_Addr2, 0x38, 0xFF); // write a 1 to CHIP_EN
  writeByte(I2C_Addr2, 0x00, 0x40); // write a 1 to CHIP_EN
  for (uint8_t i = 0x08; i <= 0x13; i++) { // start at first brightness register and go to the last
    writeByte(I2C_Addr2, i, 0x80); // write all brightness to half
  }
  for (uint8_t i = 0x14; i <= 0x37; i++) { // start at first color register and go to the last
    writeByte(I2C_Addr2, i, 0x00); // write all color to 0
  }
}

```

36, 18, 12 CHANNEL LINEAR RGB LED DRIVER

/\*\*\*\*\*\*

/\* Spin and Fade \*/

```
void mode1() {
  for (uint8_t i = 0; i < 12; i++) {
    writeByte(I2C_Addr0, 0x14 + i * 3, 0x80); // write half color to red leds
    delay(10); // add a 0.1s delay before turning on the next LED
  }
  for (uint8_t i = 0; i < 12; i++) {
    writeByte(I2C_Addr1, 0x14 + i * 3, 0x80); // write half color to red leds
    delay(10); // add a 0.1s delay before turning on the next LED
  }
  for (uint8_t i = 0; i < 12; i++) {
    writeByte(I2C_Addr2, 0x14 + i * 3, 0x80); // write half color to red leds
    delay(10); // add a 0.1s delay before turning on the next LED
  }
  for (uint8_t j = 0; j < 12; j++) {
    writeByte(I2C_Addr0, 0x16 + j * 3, 0x80); // write half color to blue leds
    delay(10);
  }
  for (uint8_t j = 0; j < 12; j++) {
    writeByte(I2C_Addr1, 0x16 + j * 3, 0x80); // write half color to blue leds
    delay(10);
  }
  for (uint8_t j = 0; j < 12; j++) {
    writeByte(I2C_Addr2, 0x16 + j * 3, 0x80); // write half color to blue leds
    delay(10);
  }
  for (uint8_t k = 0; k < 12; k++) {
    writeByte(I2C_Addr0, 0x15 + k * 3, 0x80); // write half color to green leds
    delay(10);
  }
  for (uint8_t k = 0; k < 12; k++) {
    writeByte(I2C_Addr1, 0x15 + k * 3, 0x80); // write half color to green leds
    delay(10);
  }
  for (uint8_t k = 0; k < 12; k++) {
    writeByte(I2C_Addr2, 0x15 + k * 3, 0x80); // write half color to green leds
    delay(10);
  }
  for (uint8_t m = 1; m <= 16; m++) {
    uint8_t brightness = 128 - m * 8; // start decreasing the brightness
    for (uint8_t n = 0x08; n <= 0x13; n++) {
      writeByte(I2C_Addr0, n, brightness); // write updated brightness to brightness registers
      writeByte(I2C_Addr1, n, brightness); // write updated brightness to brightness registers
      writeByte(I2C_Addr2, n, brightness); // write updated brightness to brightness registers
    }
    delay(100);
  }
  for (uint8_t p = 0x14; p <= 0x37; p++) {
    //writeByte(I2C_Addr0, p, 0x00);
    //writeByte(I2C_Addr1, p, 0x00);
    //writeByte(I2C_Addr2, p, 0x00);
    writeByte(I2C_AddrU, p, 0x00);
  }
  for (uint8_t q = 0x08; q <= 0x13; q++) {
    //writeByte(I2C_Addr0, q, 0x80);
    //writeByte(I2C_Addr1, q, 0x80);
    //writeByte(I2C_Addr2, q, 0x80);
    writeByte(I2C_AddrU, q, 0x80);
  }
  delay(500);
}
```

/\*\*\*\*\*\*

/\* Color Wheel \*/

```
int R_colors[] = {0xFF, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xFF}; // array of colors used in red LEDs
```

## 36, 18, 12 CHANNEL LINEAR RGB LED DRIVER

```
int G_colors[] = {0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00}; // array of colors used in green LEDs
int B_colors[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80}; // array of colors used in blue LEDs
```

```
void mode2() {
for (uint8_t i = 0; i < 12; i++) {
// modulo division allows us to wrap around the array without seg faulting
writeByte(I2C_Addr, 0x14, R_colors[i % 12]); // initially will access first value of R_Colors, 0xFF, then second, and so on
writeByte(I2C_Addr, 0x15, G_colors[i % 12]); // initially will access first value of G_Colors, 0x80
writeByte(I2C_Addr, 0x16, B_colors[i % 12]); // initially will access first value of B_Colors, 0x00
writeByte(I2C_Addr, 0x17, R_colors[(i + 1) % 12]); // initially will access second value of R_Colors, 0xFF, then third, and so on
writeByte(I2C_Addr, 0x18, G_colors[(i + 1) % 12]);
writeByte(I2C_Addr, 0x19, B_colors[(i + 1) % 12]);
writeByte(I2C_Addr, 0x1A, R_colors[(i + 2) % 12]);
writeByte(I2C_Addr, 0x1B, G_colors[(i + 2) % 12]);
writeByte(I2C_Addr, 0x1C, B_colors[(i + 2) % 12]);
writeByte(I2C_Addr, 0x1D, R_colors[(i + 3) % 12]);
writeByte(I2C_Addr, 0x1E, G_colors[(i + 3) % 12]);
writeByte(I2C_Addr, 0x1F, B_colors[(i + 3) % 12]);
writeByte(I2C_Addr, 0x20, R_colors[(i + 4) % 12]);
writeByte(I2C_Addr, 0x21, G_colors[(i + 4) % 12]);
writeByte(I2C_Addr, 0x22, B_colors[(i + 4) % 12]);
writeByte(I2C_Addr, 0x23, R_colors[(i + 5) % 12]);
writeByte(I2C_Addr, 0x24, G_colors[(i + 5) % 12]);
writeByte(I2C_Addr, 0x25, B_colors[(i + 5) % 12]);
writeByte(I2C_Addr, 0x26, R_colors[(i + 6) % 12]);
writeByte(I2C_Addr, 0x27, G_colors[(i + 6) % 12]);
writeByte(I2C_Addr, 0x28, B_colors[(i + 6) % 12]);
writeByte(I2C_Addr, 0x29, R_colors[(i + 7) % 12]);
writeByte(I2C_Addr, 0x2A, G_colors[(i + 7) % 12]);
writeByte(I2C_Addr, 0x2B, B_colors[(i + 7) % 12]);
writeByte(I2C_Addr, 0x2C, R_colors[(i + 8) % 12]);
writeByte(I2C_Addr, 0x2D, G_colors[(i + 8) % 12]);
writeByte(I2C_Addr, 0x2E, B_colors[(i + 8) % 12]);
writeByte(I2C_Addr, 0x2F, R_colors[(i + 9) % 12]);
writeByte(I2C_Addr, 0x30, G_colors[(i + 9) % 12]);
writeByte(I2C_Addr, 0x31, B_colors[(i + 9) % 12]);
writeByte(I2C_Addr, 0x32, R_colors[(i + 10) % 12]);
writeByte(I2C_Addr, 0x33, G_colors[(i + 10) % 12]);
writeByte(I2C_Addr, 0x34, B_colors[(i + 10) % 12]);
writeByte(I2C_Addr, 0x35, R_colors[(i + 11) % 12]);
writeByte(I2C_Addr, 0x36, G_colors[(i + 11) % 12]);
writeByte(I2C_Addr, 0x37, B_colors[(i + 11) % 12]);
delay(200);
}
}
/*****
```

```
/* Smooth */
```

```
void mode3() {
//ADDRESS 30
for (uint8_t i = 0x00; i <= 0xFE; i += 1) { // increase color value slowly
for (uint8_t g = 0x15; g <= 0x36; g += 3) { // write to all green LEDs
writeByte(I2C_Addr0, g, i);
}
for (uint8_t r = 0x14; r <= 0x35; r += 3) { // write to all red LEDs
writeByte(I2C_Addr0, r, 0xFE - i); // decrease color value slowly
}
delay(2);
}
for (uint8_t i = 0x00; i <= 0xFE; i += 1) {
for (uint8_t b = 0x16; b <= 0x37; b += 3) {
writeByte(I2C_Addr0, b, i);
}
for (uint8_t g = 0x15; g <= 0x36; g += 3) {
writeByte(I2C_Addr0, g, 0xFE - i);
}
delay(2);
}
for (uint8_t i = 0x00; i <= 0xFE; i += 1) {
```





36, 18, 12 CHANNEL LINEAR RGB LED DRIVER

```
// turn LEDs 0-4 to yellow
writeByte(I2C_Addr, 0x14, 0xFF);
writeByte(I2C_Addr, 0x15, 0xFF);
writeByte(I2C_Addr, 0x16, 0x00);
writeByte(I2C_Addr, 0x17, 0xFF);
writeByte(I2C_Addr, 0x18, 0xFF);
writeByte(I2C_Addr, 0x19, 0x00);
writeByte(I2C_Addr, 0x1A, 0xFF);
writeByte(I2C_Addr, 0x1B, 0xFF);
writeByte(I2C_Addr, 0x1C, 0x00);
writeByte(I2C_Addr, 0x1D, 0xFF);
writeByte(I2C_Addr, 0x1E, 0xFF);
writeByte(I2C_Addr, 0x1F, 0x00);
writeByte(I2C_Addr, 0x20, 0xFF);
writeByte(I2C_Addr, 0x21, 0xFF);
writeByte(I2C_Addr, 0x22, 0x00);

// turn LEDs 8-11 to yellow
writeByte(I2C_Addr, 0x2C, 0xFF);
writeByte(I2C_Addr, 0x2D, 0xFF);
writeByte(I2C_Addr, 0x2E, 0x00);
writeByte(I2C_Addr, 0x2F, 0xFF);
writeByte(I2C_Addr, 0x30, 0xFF);
writeByte(I2C_Addr, 0x31, 0x00);
writeByte(I2C_Addr, 0x32, 0xFF);
writeByte(I2C_Addr, 0x33, 0xFF);
writeByte(I2C_Addr, 0x34, 0x00);
writeByte(I2C_Addr, 0x35, 0xFF);
writeByte(I2C_Addr, 0x36, 0xFF);
writeByte(I2C_Addr, 0x37, 0x00);

// create blue pellet at LED 6
writeByte(I2C_Addr, 0x26, 0x00);
writeByte(I2C_Addr, 0x27, 0x00);
writeByte(I2C_Addr, 0x28, 0xFF);

// create the closing effect
delay(1000);
writeByte(I2C_Addr, 0x23, 0xFF);
writeByte(I2C_Addr, 0x24, 0xFF);
writeByte(I2C_Addr, 0x25, 0x00);
writeByte(I2C_Addr, 0x29, 0xFF);
writeByte(I2C_Addr, 0x2A, 0xFF);
writeByte(I2C_Addr, 0x2B, 0x00);
delay(500);
writeByte(I2C_Addr, 0x26, 0xFF);
writeByte(I2C_Addr, 0x27, 0xFF);
writeByte(I2C_Addr, 0x28, 0x00);

// create the opening effect
delay(500);
writeByte(I2C_Addr, 0x26, 0x00);
writeByte(I2C_Addr, 0x27, 0x00);
writeByte(I2C_Addr, 0x28, 0x00);
delay(500);
writeByte(I2C_Addr, 0x23, 0x00);
writeByte(I2C_Addr, 0x24, 0x00);
writeByte(I2C_Addr, 0x25, 0x00);
writeByte(I2C_Addr, 0x29, 0x00);
writeByte(I2C_Addr, 0x2A, 0x00);
writeByte(I2C_Addr, 0x2B, 0x00);
delay(500);
}
/*****

/* Snake */

void mode5() {
  // create first green dot and first apple
  writeByte(I2C_Addr, 0x14, 0x00);
```

**36, 18, 12 CHANNEL LINEAR RGB LED DRIVER**

```
writeByte(I2C_Addr, 0x15, 0xFF);  
writeByte(I2C_Addr, 0x16, 0x00);  
writeByte(I2C_Addr, 0x23, 0xFF);  
writeByte(I2C_Addr, 0x24, 0x00);  
writeByte(I2C_Addr, 0x25, 0x00);
```

```
// erase green dot to create moving effect  
delay(200);  
writeByte(I2C_Addr, 0x14, 0x00);  
writeByte(I2C_Addr, 0x15, 0x00);  
writeByte(I2C_Addr, 0x16, 0x00);  
writeByte(I2C_Addr, 0x17, 0x00);  
writeByte(I2C_Addr, 0x18, 0xFF);  
writeByte(I2C_Addr, 0x19, 0x00);
```

```
/* Repeat above process- erase last dot and move fist dot until snake hits an apple */  
delay(200);  
writeByte(I2C_Addr, 0x17, 0x00);  
writeByte(I2C_Addr, 0x18, 0x00);  
writeByte(I2C_Addr, 0x19, 0x00);  
writeByte(I2C_Addr, 0x1A, 0x00);  
writeByte(I2C_Addr, 0x1B, 0xFF);  
writeByte(I2C_Addr, 0x1C, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x1A, 0x00);  
writeByte(I2C_Addr, 0x1B, 0x00);  
writeByte(I2C_Addr, 0x1C, 0x00);  
writeByte(I2C_Addr, 0x1D, 0x00);  
writeByte(I2C_Addr, 0x1E, 0xFF);  
writeByte(I2C_Addr, 0x1F, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x1D, 0x00);  
writeByte(I2C_Addr, 0x1E, 0x00);  
writeByte(I2C_Addr, 0x1F, 0x00);  
writeByte(I2C_Addr, 0x20, 0x00);  
writeByte(I2C_Addr, 0x21, 0xFF);  
writeByte(I2C_Addr, 0x22, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x23, 0x00);  
writeByte(I2C_Addr, 0x24, 0xFF);  
writeByte(I2C_Addr, 0x25, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x14, 0xFF);  
writeByte(I2C_Addr, 0x15, 0x00);  
writeByte(I2C_Addr, 0x16, 0x00);  
writeByte(I2C_Addr, 0x1D, 0x00);  
writeByte(I2C_Addr, 0x1E, 0xFF);  
writeByte(I2C_Addr, 0x1F, 0x00);  
writeByte(I2C_Addr, 0x23, 0x00);  
writeByte(I2C_Addr, 0x24, 0x00);  
writeByte(I2C_Addr, 0x25, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x23, 0x00);  
writeByte(I2C_Addr, 0x24, 0x00);  
writeByte(I2C_Addr, 0x25, 0x00);  
writeByte(I2C_Addr, 0x20, 0x00);  
writeByte(I2C_Addr, 0x21, 0x00);  
writeByte(I2C_Addr, 0x22, 0x00);  
writeByte(I2C_Addr, 0x1A, 0x00);  
writeByte(I2C_Addr, 0x1B, 0xFF);  
writeByte(I2C_Addr, 0x1C, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x17, 0x00);  
writeByte(I2C_Addr, 0x18, 0xFF);
```

**36, 18, 12 CHANNEL LINEAR RGB LED DRIVER**

```
writeByte(I2C_Addr, 0x19, 0x00);  
writeByte(I2C_Addr, 0x1D, 0x00);  
writeByte(I2C_Addr, 0x1E, 0x00);  
writeByte(I2C_Addr, 0x1F, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x14, 0x00);  
writeByte(I2C_Addr, 0x15, 0xFF);  
writeByte(I2C_Addr, 0x16, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x2F, 0xFF);  
writeByte(I2C_Addr, 0x30, 0x00);  
writeByte(I2C_Addr, 0x31, 0x00);  
writeByte(I2C_Addr, 0x35, 0x00);  
writeByte(I2C_Addr, 0x36, 0xFF);  
writeByte(I2C_Addr, 0x37, 0x00);  
writeByte(I2C_Addr, 0x1A, 0x00);  
writeByte(I2C_Addr, 0x1B, 0x00);  
writeByte(I2C_Addr, 0x1C, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x17, 0x00);  
writeByte(I2C_Addr, 0x18, 0x00);  
writeByte(I2C_Addr, 0x19, 0x00);  
writeByte(I2C_Addr, 0x32, 0x00);  
writeByte(I2C_Addr, 0x33, 0xFF);  
writeByte(I2C_Addr, 0x34, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x2F, 0x00);  
writeByte(I2C_Addr, 0x30, 0xFF);  
writeByte(I2C_Addr, 0x31, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x29, 0xFF);  
writeByte(I2C_Addr, 0x2A, 0x00);  
writeByte(I2C_Addr, 0x2B, 0x00);  
writeByte(I2C_Addr, 0x14, 0x00);  
writeByte(I2C_Addr, 0x15, 0x00);  
writeByte(I2C_Addr, 0x16, 0x00);  
writeByte(I2C_Addr, 0x2C, 0x00);  
writeByte(I2C_Addr, 0x2D, 0xFF);  
writeByte(I2C_Addr, 0x2E, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x29, 0x00);  
writeByte(I2C_Addr, 0x2A, 0xFF);  
writeByte(I2C_Addr, 0x2B, 0x00);  
writeByte(I2C_Addr, 0x1A, 0xFF);  
writeByte(I2C_Addr, 0x1B, 0x00);  
writeByte(I2C_Addr, 0x1C, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x29, 0x00);  
writeByte(I2C_Addr, 0x2A, 0x00);  
writeByte(I2C_Addr, 0x2B, 0x00);  
writeByte(I2C_Addr, 0x14, 0x00);  
writeByte(I2C_Addr, 0x15, 0xFF);  
writeByte(I2C_Addr, 0x16, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x2B, 0x00);  
writeByte(I2C_Addr, 0x2C, 0x00);  
writeByte(I2C_Addr, 0x2D, 0x00);  
writeByte(I2C_Addr, 0x17, 0x00);  
writeByte(I2C_Addr, 0x18, 0xFF);  
writeByte(I2C_Addr, 0x19, 0x00);
```

```
delay(200);
```

**36, 18, 12 CHANNEL LINEAR RGB LED DRIVER**

```
writeByte(I2C_Addr, 0x1A, 0x00);  
writeByte(I2C_Addr, 0x1B, 0xFF);  
writeByte(I2C_Addr, 0x1C, 0x00);  
writeByte(I2C_Addr, 0x23, 0xFF);  
writeByte(I2C_Addr, 0x24, 0x00);  
writeByte(I2C_Addr, 0x25, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x1D, 0x00);  
writeByte(I2C_Addr, 0x1E, 0xFF);  
writeByte(I2C_Addr, 0x1F, 0x00);  
writeByte(I2C_Addr, 0x2F, 0x00);  
writeByte(I2C_Addr, 0x30, 0x00);  
writeByte(I2C_Addr, 0x31, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x20, 0x00);  
writeByte(I2C_Addr, 0x21, 0xFF);  
writeByte(I2C_Addr, 0x22, 0x00);  
writeByte(I2C_Addr, 0x32, 0x00);  
writeByte(I2C_Addr, 0x33, 0x00);  
writeByte(I2C_Addr, 0x34, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x23, 0x00);  
writeByte(I2C_Addr, 0x24, 0xFF);  
writeByte(I2C_Addr, 0x25, 0x00);  
writeByte(I2C_Addr, 0x29, 0xFF);  
writeByte(I2C_Addr, 0x2A, 0x00);  
writeByte(I2C_Addr, 0x2B, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x26, 0x00);  
writeByte(I2C_Addr, 0x27, 0xFF);  
writeByte(I2C_Addr, 0x28, 0x00);  
writeByte(I2C_Addr, 0x35, 0x00);  
writeByte(I2C_Addr, 0x36, 0x00);  
writeByte(I2C_Addr, 0x37, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x29, 0x00);  
writeByte(I2C_Addr, 0x2A, 0xFF);  
writeByte(I2C_Addr, 0x2B, 0x00);  
writeByte(I2C_Addr, 0x2F, 0xFF);  
writeByte(I2C_Addr, 0x30, 0x00);  
writeByte(I2C_Addr, 0x31, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x14, 0x00);  
writeByte(I2C_Addr, 0x15, 0x00);  
writeByte(I2C_Addr, 0x16, 0x00);  
writeByte(I2C_Addr, 0x2C, 0x00);  
writeByte(I2C_Addr, 0x2D, 0xFF);  
writeByte(I2C_Addr, 0x2E, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x2F, 0x00);  
writeByte(I2C_Addr, 0x30, 0xFF);  
writeByte(I2C_Addr, 0x31, 0x00);  
writeByte(I2C_Addr, 0x14, 0xFF);  
writeByte(I2C_Addr, 0x15, 0x00);  
writeByte(I2C_Addr, 0x16, 0x00);
```

```
delay(200);  
writeByte(I2C_Addr, 0x17, 0x00);  
writeByte(I2C_Addr, 0x18, 0x00);  
writeByte(I2C_Addr, 0x19, 0x00);  
writeByte(I2C_Addr, 0x32, 0x00);  
writeByte(I2C_Addr, 0x33, 0xFF);  
writeByte(I2C_Addr, 0x34, 0x00);
```

```

delay(200);
writeByte(I2C_Addr, 0x1A, 0x00);
writeByte(I2C_Addr, 0x1B, 0x00);
writeByte(I2C_Addr, 0x1C, 0x00);
writeByte(I2C_Addr, 0x35, 0x00);
writeByte(I2C_Addr, 0x36, 0xFF);
writeByte(I2C_Addr, 0x37, 0x00);

delay(200);
writeByte(I2C_Addr, 0x14, 0x00);
writeByte(I2C_Addr, 0x15, 0xFF);
writeByte(I2C_Addr, 0x16, 0x00);
writeByte(I2C_Addr, 0x1A, 0xFF);
writeByte(I2C_Addr, 0x1B, 0x00);
writeByte(I2C_Addr, 0x1C, 0x00);

delay(200);
writeByte(I2C_Addr, 0x1D, 0x00);
writeByte(I2C_Addr, 0x1E, 0x00);
writeByte(I2C_Addr, 0x1F, 0x00);
writeByte(I2C_Addr, 0x17, 0x00);
writeByte(I2C_Addr, 0x18, 0xFF);
writeByte(I2C_Addr, 0x19, 0x00);

delay(200);
writeByte(I2C_Addr, 0x1A, 0x00);
writeByte(I2C_Addr, 0x1B, 0xFF);
writeByte(I2C_Addr, 0x1C, 0x00);
writeByte(I2C_Addr, 0x1D, 0xFF);
writeByte(I2C_Addr, 0x1E, 0x00);
writeByte(I2C_Addr, 0x1F, 0x00);

delay(200);
writeByte(I2C_Addr, 0x1D, 0x00);
writeByte(I2C_Addr, 0x1E, 0xFF);
writeByte(I2C_Addr, 0x1F, 0x00);

// Victory signal
delay(1000);
for (int i = 0; i < 3; i++) {
    for (uint8_t r = 0x14; r <= 0x37; r++) {
        writeByte(I2C_Addr, r, 0xFF);
    }
    delay(250);
    for (uint8_t r = 0x14; r <= 0x37; r++) {
        writeByte(I2C_Addr, r, 0x00);
    }
    delay(250);
}
delay(1000);
}
/*****

/* Colorful Spinner */

void mode6() {
int R_colors[] = {0xFF, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xFF};
int G_colors[] = {0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00};
int B_colors[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80};
int brightnesses[] = {0xF9, 0xE2, 0xCC, 0xB7, 0xA0, 0x8D, 0x80, 0x62, 0x50, 0x35, 0x1D, 0x10}; // array of brightness values

writeByte(I2C_Addr, 0x14, R_colors[0]);
writeByte(I2C_Addr, 0x15, G_colors[0]);
writeByte(I2C_Addr, 0x16, B_colors[0]);
writeByte(I2C_Addr, 0x17, R_colors[1]);
writeByte(I2C_Addr, 0x18, G_colors[1]);
writeByte(I2C_Addr, 0x19, B_colors[1]);
writeByte(I2C_Addr, 0x1A, R_colors[2]);

```

36, 18, 12 CHANNEL LINEAR RGB LED DRIVER

```

writeByte(I2C_Addr, 0x1B, G_colors[2]);
writeByte(I2C_Addr, 0x1C, B_colors[2]);
writeByte(I2C_Addr, 0x1D, R_colors[3]);
writeByte(I2C_Addr, 0x1E, G_colors[3]);
writeByte(I2C_Addr, 0x1F, B_colors[3]);
writeByte(I2C_Addr, 0x20, R_colors[4]);
writeByte(I2C_Addr, 0x21, G_colors[4]);
writeByte(I2C_Addr, 0x22, B_colors[4]);
writeByte(I2C_Addr, 0x23, R_colors[5]);
writeByte(I2C_Addr, 0x24, G_colors[5]);
writeByte(I2C_Addr, 0x25, B_colors[5]);
writeByte(I2C_Addr, 0x26, R_colors[6]);
writeByte(I2C_Addr, 0x27, G_colors[6]);
writeByte(I2C_Addr, 0x28, B_colors[6]);
writeByte(I2C_Addr, 0x29, R_colors[7]);
writeByte(I2C_Addr, 0x2A, G_colors[7]);
writeByte(I2C_Addr, 0x2B, B_colors[7]);
writeByte(I2C_Addr, 0x2C, R_colors[8]);
writeByte(I2C_Addr, 0x2D, G_colors[8]);
writeByte(I2C_Addr, 0x2E, B_colors[8]);
writeByte(I2C_Addr, 0x2F, R_colors[9]);
writeByte(I2C_Addr, 0x30, G_colors[9]);
writeByte(I2C_Addr, 0x31, B_colors[9]);
writeByte(I2C_Addr, 0x32, R_colors[10]);
writeByte(I2C_Addr, 0x33, G_colors[10]);
writeByte(I2C_Addr, 0x34, B_colors[10]);
writeByte(I2C_Addr, 0x35, R_colors[11]);
writeByte(I2C_Addr, 0x36, G_colors[11]);
writeByte(I2C_Addr, 0x37, B_colors[11]);
for (uint8_t i = 0; i < 12; i++) {
    writeByte(I2C_Addr, 0x08, brightnesses[i % 12]);
    writeByte(I2C_Addr, 0x09, brightnesses[(i + 1) % 12]);
    writeByte(I2C_Addr, 0x0A, brightnesses[(i + 2) % 12]);
    writeByte(I2C_Addr, 0x0B, brightnesses[(i + 3) % 12]);
    writeByte(I2C_Addr, 0x0C, brightnesses[(i + 4) % 12]);
    writeByte(I2C_Addr, 0x0D, brightnesses[(i + 5) % 12]);
    writeByte(I2C_Addr, 0x0E, brightnesses[(i + 6) % 12]);
    writeByte(I2C_Addr, 0x0F, brightnesses[(i + 7) % 12]);
    writeByte(I2C_Addr, 0x10, brightnesses[(i + 8) % 12]);
    writeByte(I2C_Addr, 0x11, brightnesses[(i + 9) % 12]);
    writeByte(I2C_Addr, 0x12, brightnesses[(i + 10) % 12]);
    writeByte(I2C_Addr, 0x13, brightnesses[(i + 11) % 12]);
    delay(175);
}
}
/*****

/* Red Spinner */

void mode7() {
    for (uint8_t i = 0; i < 12; i++) {
        writeByte(I2C_Addr, 0x14 + i * 3, 0x80);
    }
    int brightnesses[] = {0xF9, 0xE2, 0xCC, 0xB7, 0xA0, 0x8D, 0x80, 0x62, 0x50, 0x35, 0x1D, 0x10};
    for (uint8_t i = 0; i < 12; i++) {
        writeByte(I2C_Addr, 0x08, brightnesses[i % 12]);
        writeByte(I2C_Addr, 0x09, brightnesses[(i + 1) % 12]);
        writeByte(I2C_Addr, 0x0A, brightnesses[(i + 2) % 12]);
        writeByte(I2C_Addr, 0x0B, brightnesses[(i + 3) % 12]);
        writeByte(I2C_Addr, 0x0C, brightnesses[(i + 4) % 12]);
        writeByte(I2C_Addr, 0x0D, brightnesses[(i + 5) % 12]);
        writeByte(I2C_Addr, 0x0E, brightnesses[(i + 6) % 12]);
        writeByte(I2C_Addr, 0x0F, brightnesses[(i + 7) % 12]);
        writeByte(I2C_Addr, 0x10, brightnesses[(i + 8) % 12]);
        writeByte(I2C_Addr, 0x11, brightnesses[(i + 9) % 12]);
        writeByte(I2C_Addr, 0x12, brightnesses[(i + 10) % 12]);
        writeByte(I2C_Addr, 0x13, brightnesses[(i + 11) % 12]);
        delay(175);
    }
}
}

```

/\*\*\*\*\*

/\* Newton's Cradle \*/

```
void mode8() {
  for (uint8_t i = 0; i < 12; i++) {
    writeByte(I2C_Addr, 0x14 + i * 3, 0xFF);
  }
  for (uint8_t j = 0; j < 12; j++) {
    writeByte(I2C_Addr, 0x16 + j * 3, 0x80);
  }
  for (uint8_t k = 0; k < 12; k++) {
    writeByte(I2C_Addr, 0x15 + k * 3, 0x00);
  }
}
```

// use increased delay values to create illusion of gravity- 'higher' the LEDs get, the harder it is to 'lift' them

// lift up one side

```
delay(100);
writeByte(I2C_Addr, 0x23, 0x00);
writeByte(I2C_Addr, 0x24, 0x00);
writeByte(I2C_Addr, 0x25, 0x00);
delay(250);
writeByte(I2C_Addr, 0x20, 0x00);
writeByte(I2C_Addr, 0x21, 0x00);
writeByte(I2C_Addr, 0x22, 0x00);
delay(375);
writeByte(I2C_Addr, 0x1D, 0x00);
writeByte(I2C_Addr, 0x1E, 0x00);
writeByte(I2C_Addr, 0x1F, 0x00);
delay(500);
writeByte(I2C_Addr, 0x1A, 0x00);
writeByte(I2C_Addr, 0x1B, 0x00);
writeByte(I2C_Addr, 0x1C, 0x00);
delay(725);
writeByte(I2C_Addr, 0x17, 0x00);
writeByte(I2C_Addr, 0x18, 0x00);
writeByte(I2C_Addr, 0x19, 0x00);
```

// dropping down same side

```
delay(725);
writeByte(I2C_Addr, 0x17, 0xFF);
writeByte(I2C_Addr, 0x18, 0x00);
writeByte(I2C_Addr, 0x19, 0x80);
delay(500);
writeByte(I2C_Addr, 0x1A, 0xFF);
writeByte(I2C_Addr, 0x1B, 0x00);
writeByte(I2C_Addr, 0x1C, 0x80);
delay(375);
writeByte(I2C_Addr, 0x1D, 0xFF);
writeByte(I2C_Addr, 0x1E, 0x00);
writeByte(I2C_Addr, 0x1F, 0x80);
delay(250);
writeByte(I2C_Addr, 0x20, 0xFF);
writeByte(I2C_Addr, 0x21, 0x00);
writeByte(I2C_Addr, 0x22, 0x80);
delay(125);
writeByte(I2C_Addr, 0x23, 0xFF);
writeByte(I2C_Addr, 0x24, 0x00);
writeByte(I2C_Addr, 0x25, 0x80);
```

// lift up other side

```
delay(100);
writeByte(I2C_Addr, 0x26, 0x00);
writeByte(I2C_Addr, 0x27, 0x00);
writeByte(I2C_Addr, 0x28, 0x00);
delay(200);
writeByte(I2C_Addr, 0x29, 0x00);
writeByte(I2C_Addr, 0x2A, 0x00);
writeByte(I2C_Addr, 0x2B, 0x00);
```



36, 18, 12 CHANNEL LINEAR RGB LED DRIVER

```

delay(350);
writeByte(I2C_Addr, 0x2C, 0x00);
writeByte(I2C_Addr, 0x2D, 0x00);
writeByte(I2C_Addr, 0x2E, 0x00);
delay(450);
writeByte(I2C_Addr, 0x2F, 0x00);
writeByte(I2C_Addr, 0x30, 0x00);
writeByte(I2C_Addr, 0x31, 0x00);
delay(650);
writeByte(I2C_Addr, 0x32, 0x00);
writeByte(I2C_Addr, 0x33, 0x00);
writeByte(I2C_Addr, 0x34, 0x00);

// dropping down other side
delay(650);
writeByte(I2C_Addr, 0x32, 0xFF);
writeByte(I2C_Addr, 0x33, 0x00);
writeByte(I2C_Addr, 0x34, 0x80);
delay(500);
writeByte(I2C_Addr, 0x2F, 0xFF);
writeByte(I2C_Addr, 0x30, 0x00);
writeByte(I2C_Addr, 0x31, 0x80);
delay(375);
writeByte(I2C_Addr, 0x2C, 0xFF);
writeByte(I2C_Addr, 0x2D, 0x00);
writeByte(I2C_Addr, 0x2E, 0x80);
delay(250);
writeByte(I2C_Addr, 0x29, 0xFF);
writeByte(I2C_Addr, 0x2A, 0x00);
writeByte(I2C_Addr, 0x2B, 0x80);
delay(125);
writeByte(I2C_Addr, 0x26, 0xFF);
writeByte(I2C_Addr, 0x27, 0x00);
writeByte(I2C_Addr, 0x28, 0x80);
}
/*****/

/* Time Bomb */

void mode9() {
for (int i = 500; i >= 0; i -= 20) // slowly decrease time between blinking on and off
delay(i);
for (uint8_t i = 0; i < 12; i++) {
writeByte(I2C_Addr, 0x14 + i * 3, 0xFF);
}
delay(i);
for (uint8_t i = 0; i < 12; i++) {
writeByte(I2C_Addr, 0x14 + i * 3, 0x00);
}
}
delay(1000);
}
/*****/

/* 4th of July */

void mode10() {
int R_colors[] = {0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0x00};
int G_colors[] = {0x00, 0xFF, 0x00, 0x00, 0xFF, 0x00, 0x00, 0xFF, 0x00, 0x00, 0xFF, 0x00};
int B_colors[] = {0x00, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0xFF};

for (uint8_t i = 0; i < 12; i++) {
writeByte(I2C_Addr, 0x14, R_colors[i % 12]);
writeByte(I2C_Addr, 0x15, G_colors[i % 12]);
writeByte(I2C_Addr, 0x16, B_colors[i % 12]);
writeByte(I2C_Addr, 0x17, R_colors[(i + 1) % 12]);
writeByte(I2C_Addr, 0x18, G_colors[(i + 1) % 12]);
writeByte(I2C_Addr, 0x19, B_colors[(i + 1) % 12]);
writeByte(I2C_Addr, 0x1A, R_colors[(i + 2) % 12]);
writeByte(I2C_Addr, 0x1B, G_colors[(i + 2) % 12]);
}
}

```

36, 18, 12 CHANNEL LINEAR RGB LED DRIVER

```

writeByte(I2C_Addr, 0x1C, B_colors[(i + 2) % 12]);
writeByte(I2C_Addr, 0x1D, R_colors[(i + 3) % 12]);
writeByte(I2C_Addr, 0x1E, G_colors[(i + 3) % 12]);
writeByte(I2C_Addr, 0x1F, B_colors[(i + 3) % 12]);
writeByte(I2C_Addr, 0x20, R_colors[(i + 4) % 12]);
writeByte(I2C_Addr, 0x21, G_colors[(i + 4) % 12]);
writeByte(I2C_Addr, 0x22, B_colors[(i + 4) % 12]);
writeByte(I2C_Addr, 0x23, R_colors[(i + 5) % 12]);
writeByte(I2C_Addr, 0x24, G_colors[(i + 5) % 12]);
writeByte(I2C_Addr, 0x25, B_colors[(i + 5) % 12]);
writeByte(I2C_Addr, 0x26, R_colors[(i + 6) % 12]);
writeByte(I2C_Addr, 0x27, G_colors[(i + 6) % 12]);
writeByte(I2C_Addr, 0x28, B_colors[(i + 6) % 12]);
writeByte(I2C_Addr, 0x29, R_colors[(i + 7) % 12]);
writeByte(I2C_Addr, 0x2A, G_colors[(i + 7) % 12]);
writeByte(I2C_Addr, 0x2B, B_colors[(i + 7) % 12]);
writeByte(I2C_Addr, 0x2C, R_colors[(i + 8) % 12]);
writeByte(I2C_Addr, 0x2D, G_colors[(i + 8) % 12]);
writeByte(I2C_Addr, 0x2E, B_colors[(i + 8) % 12]);
writeByte(I2C_Addr, 0x2F, R_colors[(i + 9) % 12]);
writeByte(I2C_Addr, 0x30, G_colors[(i + 9) % 12]);
writeByte(I2C_Addr, 0x31, B_colors[(i + 9) % 12]);
writeByte(I2C_Addr, 0x32, R_colors[(i + 10) % 12]);
writeByte(I2C_Addr, 0x33, G_colors[(i + 10) % 12]);
writeByte(I2C_Addr, 0x34, B_colors[(i + 10) % 12]);
writeByte(I2C_Addr, 0x35, R_colors[(i + 11) % 12]);
writeByte(I2C_Addr, 0x36, G_colors[(i + 11) % 12]);
writeByte(I2C_Addr, 0x37, B_colors[(i + 11) % 12]);
delay(350);
}
for (int i = 0; i < 3; i++) { // blink leds three times
delay(300);
for (uint8_t j = 0x08; j <= 0x13; j++) {
writeByte(I2C_Addr, j, 0x80);
}
delay(300);
for (uint8_t j = 0x08; j <= 0x13; j++) {
writeByte(I2C_Addr, j, 0x00);
}
}
delay(300);
// firework show
for (int i = 0; i < 12; i++) {
uint8_t temp = random(0x08, 0x14); // random brightness register selection
writeByte(I2C_Addr, temp, 0x80); // turn the selected LED to half brightness
for (uint8_t m = 1; m <= 16; m++) {
uint8_t brightness = 128 - m * 8; // start decreasing the brightness
writeByte(I2C_Addr, temp, brightness);
delay(50);
}
delay(100);
}
for (uint8_t q = 0x08; q <= 0x13; q++) {
writeByte(I2C_Addr, q, 0x80);
}
delay(500);
}
/*****

/* Heartbeat */

void mode11() {
for (uint8_t i = 0; i < 12; i++) {
writeByte(I2C_Addr, 0x14 + i * 3, 0xFF);
}
for (uint8_t j = 0; j < 12; j++) {
writeByte(I2C_Addr, 0x16 + j * 3, 0x80);
}
for (uint8_t k = 0; k < 12; k++) {
writeByte(I2C_Addr, 0x15 + k * 3, 0x00);
}
}

```

```

}
for (uint8_t m = 1; m <= 12; m++) {
  uint8_t brightness = 128 - m * 8;
  for (uint8_t q = 0x08; q <= 0x13; q++) {
    writeByte(I2C_Addr, q, brightness);
  }
  delay(20);
}
delay(25);
for (uint8_t m = 1; m <= 12; m++) {
  uint8_t brightness = 128 - m * 8;
  for (uint8_t q = 0x08; q <= 0x13; q++) {
    writeByte(I2C_Addr, q, brightness);
  }
  delay(50);
}
delay(250);
}

void mode12() {
  uint8_t t = 250;
  writeByte(I2C_Addr, 0x01, 0x86); // Set EN_LOG to OFF
  writeByte(I2C_Addr, 0x02, 0xFF); // Set ALL RGB to Bank mode
  writeByte(I2C_Addr, 0x03, 0x0F); // Set ALL RGB to Bank mode
  writeByte(I2C_Addr, 0x04, 0xf0); // Set Bank Brightness to medium
  writeByte(I2C_Addr, 0x05, 0x00); // SET BANK A Color to 0
  writeByte(I2C_Addr, 0x06, 0x00); // SET BANK B Color to 0
  writeByte(I2C_Addr, 0x07, 0x00); // SET BANK C Color to 0
  delay(200);

  for (uint8_t r = 0; r <= t; r++) {
    writeByte(I2C_Addr, 0x05, r); //analogWrite(bluePin,blue)
    delay(0);
    for (uint8_t g = 0; g <= t; g++) {
      writeByte(I2C_Addr, 0x06, g); //analogWrite(greenPin,green)
      delay(0);
      for (uint8_t b = 0; b <= t; b++) {
        writeByte(I2C_Addr, 0x07, b); //analogWrite(redPin,red)
        delay(0);
      }
    }
  }
}

void mode13() {
  uint8_t t = 0x32;
  uint8_t TIME = 40;
  // writeByte(I2C_Addr, 0x01, 0xA6); // Set EN_LOG to OFF
  writeByte(I2C_Addr, 0x02, 0xFF); // Set ALL RGB to Bank mode
  writeByte(I2C_Addr, 0x03, 0x0F); // Set ALL RGB to Bank mode
  writeByte(I2C_Addr, 0x04, 0xf0); // Set Bank Brightness to medium
  writeByte(I2C_Addr, 0x05, t); // SET BANK A Color to FF
  writeByte(I2C_Addr, 0x06, 0x00); // SET BANK B Color to 0
  writeByte(I2C_Addr, 0x07, 0x00); // SET BANK C Color to 0
  delay(50);

  for (uint8_t r = 0x00; r <= t; r++)
  {
    uint8_t m = t - r;
    writeByte(I2C_Addr, 0x05, m); // red from 255 to 0
    writeByte(I2C_Addr, 0x07, r); // blue concurrently from 0 to 255
    delay(TIME);
  }
  for (uint8_t g = 0x00; g <= t; g++)
  {
    uint8_t m = t - g;
    writeByte(I2C_Addr, 0x07, m); // blue from 255 to 0
    writeByte(I2C_Addr, 0x06, g); //green from 0 to 255 concurrently
    delay(TIME);
  }
}

```

36, 18, 12 CHANNEL LINEAR RGB LED DRIVER

```

for (uint8_t b = 0; b <= t; b++)
{
    uint8_t m = t - b;
    writeByte(I2C_Addr, 0x06, m); //Green from 255 to 0
    writeByte(I2C_Addr, 0x05, b); // red from 0 to 255 concurrently
    delay(TIME);
}
}

void mode14() {
    uint8_t t = 0x32;
    uint8_t TIME = 40;
    // writeByte(I2C_Addr, 0x01, 0xA6); // Set EN_LOG to OFF
    writeByte(I2C_Addr, 0x02, 0xFF); // Set ALL RGB to Bank mode
    writeByte(I2C_Addr, 0x03, 0x0F); // Set ALL RGB to Bank mode
    writeByte(I2C_Addr, 0x04, 0xF0); // Set Bank Brightness to medium
    writeByte(I2C_Addr, 0x05, t); // SET BANK A Color to FF
    writeByte(I2C_Addr, 0x06, 0x00); // SET BANK B Color to 0
    writeByte(I2C_Addr, 0x07, 0x00); // SET BANK C Color to 0
    delay(50);

    for (uint8_t r = 0x00; r <= t; r++)
    {
        uint8_t m = t - r;
        // writeByte(I2C_Addr, 0x05, m); //
        writeByte(I2C_Addr, 0x05, r+7); // red
        delay(TIME);
    }
    // for (uint8_t g = 0x00; g <= t; g++)
    // {
    //     uint8_t m = t - g;
    //     writeByte(I2C_Addr, 0x07, m); // blue from 255 to 0
    //     writeByte(I2C_Addr, 0x06, g); //green from 0 to 255 concurrently
    //     delay(TIME);
    // }
    // for (uint8_t b = 0; b <= t; b++)
    // {
    //     uint8_t m = t - b;
    //     writeByte(I2C_Addr, 0x06, m); //Green from 255 to 0
    //     writeByte(I2C_Addr, 0x05, b); // red from 0 to 255 concurrently
    //     delay(TIME);
    // }
    // }
}
/*****/

```

**IMPORTANT NOTICE**

1. DIODES INCORPORATED (Diodes) AND ITS SUBSIDIARIES MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARDS TO ANY INFORMATION CONTAINED IN THIS DOCUMENT, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION).
2. The Information contained herein is for informational purpose only and is provided only to illustrate the operation of Diodes' products described herein and application examples. Diodes does not assume any liability arising out of the application or use of this document or any product described herein. This document is intended for skilled and technically trained engineering customers and users who design with Diodes' products. Diodes' products may be used to facilitate safety-related applications; however, in all instances customers and users are responsible for (a) selecting the appropriate Diodes products for their applications, (b) evaluating the suitability of Diodes' products for their intended applications, (c) ensuring their applications, which incorporate Diodes' products, comply the applicable legal and regulatory requirements as well as safety and functional-safety related standards, and (d) ensuring they design with appropriate safeguards (including testing, validation, quality control techniques, redundancy, malfunction prevention, and appropriate treatment for aging degradation) to minimize the risks associated with their applications.
3. Diodes assumes no liability for any application-related information, support, assistance or feedback that may be provided by Diodes from time to time. Any customer or user of this document or products described herein will assume all risks and liabilities associated with such use, and will hold Diodes and all companies whose products are represented herein or on Diodes' websites, harmless against all damages and liabilities.
4. Products described herein may be covered by one or more United States, international or foreign patents and pending patent applications. Product names and markings noted herein may also be covered by one or more United States, international or foreign trademarks and trademark applications. Diodes does not convey any license under any of its intellectual property rights or the rights of any third parties (including third parties whose products and services may be described in this document or on Diodes' website) under this document.
5. Diodes' products are provided subject to Diodes' Standard Terms and Conditions of Sale (<https://www.diodes.com/about/company/terms-and-conditions/terms-and-conditions-of-sales/>) or other applicable terms. This document does not alter or expand the applicable warranties provided by Diodes. Diodes does not warrant or accept any liability whatsoever in respect of any products purchased through unauthorized sales channel.
6. Diodes' products and technology may not be used for or incorporated into any products or systems whose manufacture, use or sale is prohibited under any applicable laws and regulations. Should customers or users use Diodes' products in contravention of any applicable laws or regulations, or for any unintended or unauthorized application, customers and users will (a) be solely responsible for any damages, losses or penalties arising in connection therewith or as a result thereof, and (b) indemnify and hold Diodes and its representatives and agents harmless against any and all claims, damages, expenses, and attorney fees arising out of, directly or indirectly, any claim relating to any noncompliance with the applicable laws and regulations, as well as any unintended or unauthorized application.
7. While efforts have been made to ensure the information contained in this document is accurate, complete and current, it may contain technical inaccuracies, omissions and typographical errors. Diodes does not warrant that information contained in this document is error-free and Diodes is under no obligation to update or otherwise correct this information. Notwithstanding the foregoing, Diodes reserves the right to make modifications, enhancements, improvements, corrections or other changes without further notice to this document and any product described herein. This document is written in English but may be translated into multiple languages for reference. Only the English version of this document is the final and determinative format released by Diodes.
8. Any unauthorized copying, modification, distribution, transmission, display or other use of this document (or any portion hereof) is prohibited. Diodes assumes no responsibility for any losses incurred by the customers or users or any third parties arising from any such unauthorized use.
9. This Notice may be periodically updated with the most recent version available at <https://www.diodes.com/about/company/terms-and-conditions/important-notice>

The Diodes logo is a registered trademark of Diodes Incorporated in the United States and other countries.  
All other trademarks are the property of their respective owners.  
© 2024 Diodes Incorporated. All Rights Reserved.

[www.diodes.com](http://www.diodes.com)