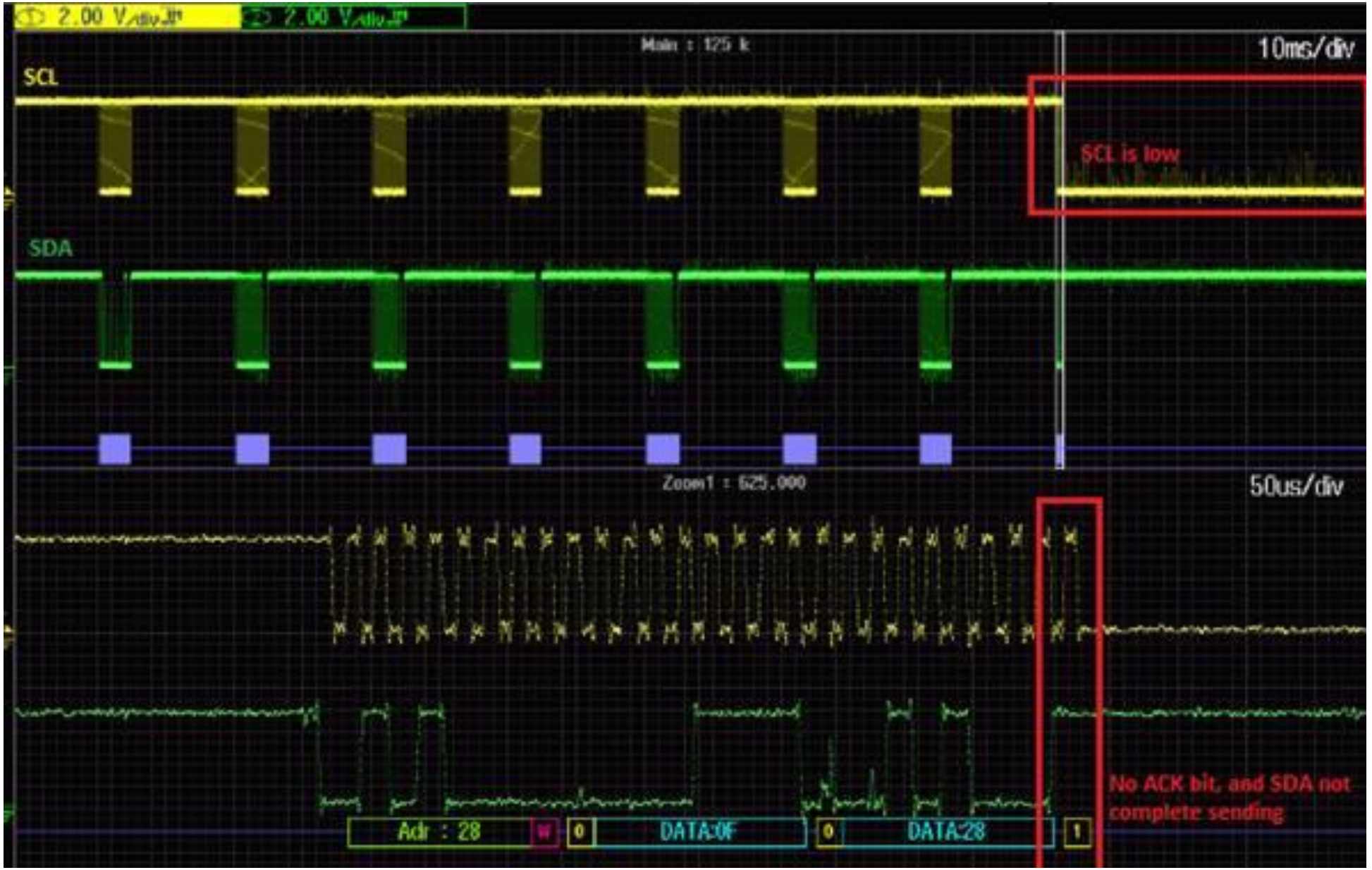

(waveform 1): SCL is low, SDA stop sending. For this case, there is no ACK bit (0), so there is possibility that LED driver side is busy or having problem hence could not acknowledge the I2C data.

(waveform 2): SDA is not complete sending, SDA is pulled low with ripple.
For this case, we need to know why the SDA is pulled low by LED driver.

I2C Data

1) Based on I2C data, host will send Chip_EN after AC ON for awhile.

2) After that, host write I2C (23 bytes of I2C data) every 10msec.

ndex	m:s.ms.us	Dur	Len	Err	S/P	Addr	Record	Data
0	0:00.000.000						Capture started	[03/18/19 20:51:05]
1	0:01.333.063	355 ms	2 B	T...		N...	Corrupt Transact...	FF* C0*
2	0:02.355.237	294 us	2 B		SP	28	Write Transaction	00 40
3	0:02.363.574	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4	0:02.373.237	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5	0:02.383.202	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
6	0:02.393.207	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7	0:02.403.200	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
8	0:02.413.207	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
9	0:02.423.199	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10	0:02.433.215	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
11	0:02.443.205	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12	0:02.453.216	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
13	0:02.463.207	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
14	0:02.473.215	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
15	0:02.483.519	2.33 ms	24 B		SP	28	Write Transaction	0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16	0:02.493.322	2.33 ms	24 B		SP	28	Write Transaction	0F FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00 00
17	0:02.503.273	2.33 ms	24 B		SP	28	Write Transaction	0F FF 02 00 FF 02 00 FF 02 00 FF 02 00 FF 02 00 FF 02 00 00 00 00 00 00
18	0:02.513.278	2.33 ms	24 B		SP	28	Write Transaction	0F FF 07 00 FF 07 00 FF 07 00 FF 07 00 FF 07 00 FF 07 00 00 00 00 00 00

Chip_EN

Write 23 bytes of data

I2C write for every 10msec

3) When Party Light turn on, the data keep changing frequently.
4) After about 1 ~ 2 mins, then the writing process failed.

Data changed every 10 msec

ndex	m:s.ms.us	Dur	Len	Err	S/P	Addr	Record	Data
9283	0:46.169.427	2.33 ms	24 B		SP	28	Write Transaction	0F 7F FF FF 7F FF FF 7F FF FF 7F FF FF 7F FF FF 7F FF FF FF 00 00 00 00
9284	0:46.179.439	2.33 ms	24 B		SP	28	Write Transaction	0F 7A FF FF 7A FF FF 7A FF FF 7A FF FF 7A FF FF 7A FF FF FF 00 00 00 00
9285	0:46.189.430	2.33 ms	24 B		SP	28	Write Transaction	0F 75 FF FF 75 FF FF 75 FF FF 75 FF FF 75 FF FF 75 FF FF FF 00 00 00 00
9286	0:46.199.442	2.33 ms	24 B		SP	28	Write Transaction	0F 70 FF FF 70 FF FF 70 FF FF 70 FF FF 70 FF FF 70 FF FF FF 00 00 00 00
9287	0:46.209.433	2.33 ms	24 B		SP	28	Write Transaction	0F 6B FF FF 6B FF FF 6B FF FF 6B FF FF 6B FF FF 6B FF FF FF 00 00 00 00
9288	0:46.219.445	2.33 ms	24 B		SP	28	Write Transaction	0F 66 FF FF 66 FF FF 66 FF FF 66 FF FF 66 FF FF 66 FF FF FF 00 00 00 00
9289	0:46.229.433	2.33 ms	24 B		SP	28	Write Transaction	0F 63 FF FF 63 FF FF 63 FF FF 63 FF FF 63 FF FF 63 FF FF FF 00 00 00 00
9290	0:46.239.441	2.33 ms	24 B		SP	28	Write Transaction	0F 5E FF FF 5E FF FF 5E FF FF 5E FF FF 5E FF FF 5E FF FF FF 00 00 00 00
9291	0:46.249.437	2.33 ms	24 B		SP	28	Write Transaction	0F 59 FF FF 59 FF FF 59 FF FF 59 FF FF 59 FF FF 59 FF FF FF 00 00 00 00
9292	0:46.259.452	2.33 ms	24 B		SP	28	Write Transaction	0F 54 FF FF 54 FF FF 54 FF FF 54 FF FF 54 FF FF 54 FF FF FF 00 00 00 00
9293	0:46.269.442	2.33 ms	24 B		SP	28	Write Transaction	0F 4F FF FF 4F FF FF 4F FF FF 4F FF FF 4F FF FF 4F FF FF FF 00 00 00 00
9294	0:46.279.454	2.33 ms	24 B		SP	28	Write Transaction	0F 4C FF FF 4C FF FF 4C FF FF 4C FF FF 4C FF FF 4C FF FF FF 00 00 00 00
9295	0:46.289.445	2.33 ms	24 B		SP	28	Write Transaction	0F 47 FF FF 47 FF FF 47 FF FF 47 FF FF 47 FF FF 47 FF FF FF 00 00 00 00
9296	0:46.299.453	2.33 ms	24 B		SP	28	Write Transaction	0F 42 FF FF 42 FF FF 42 FF FF 42 FF FF 42 FF FF 42 FF FF FF 00 00 00 00
9297	0:46.309.445	2.33 ms	24 B		SP	28	Write Transaction	0F 3D FF FF 3D FF FF 3D FF FF 3D FF FF 3D FF FF 3D FF FF FF 00 00 00 00
9298	0:46.319.454	2.33 ms	24 B		SP	28	Write Transaction	0F 38 FF FF 38 FF FF 38 FF FF 38 FF FF 38 FF FF 38 FF FF FF 00 00 00 00
9299	0:46.329.450	80.0 ms	7 B	P7	SP	28	Write Transaction	0F 33 FF FF 33 FF FE*
9300	0:46.409.490	257 ms	0 B		SP	N...	Corrupt Transact...	
9301	0:46.667.352	4.90 ms	0 B		SP	N...	Corrupt Transact...	

```
static  void    _mode_close_to_open(void )
{

    switch(_status.Phase){
        default:
            _status.Phase = 0;
            _led_iic.state = LED_I2C_READY;
            _led_iic.timeout = LED_I2C_TIMEOUT;
            //FALLTHROUGH
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
       case 10:
       case 11:
       case 12:
       case 13:
       case 14:
       case 15:
       case 16:
       case 17:
       case 18:
       case 19:
       case 20:
           break;
       case 21:
           /*LP5024 driver Chip_EN Enable*/
           _ledout_value = 0x40;
           _led_iic_tx(0x00, 1, &_ledout_value);
           break;
```

This function starts after AC In.
After some delay, it sends I2C data via API.

API used to send I2C.
_led_iic_tx(Register Address, size in byte, value).
It’s sending for Chip_EN.

```
/*****************************************************************************
static  void    _mode_refresh(void )
{
    unsigned char i;

#if defined(LED_REFRESH_10ms)

    /*LP5024 LED DRIVER*/

    for(i=18; i<23; i++){
        _i2c_drv[i] = (_i2c_drv[i] > 127u) ? 0xFF : 0x00;
    }

    _i2c_3ms_count++;
    if(_i2c_3ms_count >= 1){
        if (_led_iic.state == LED_I2C_READY){
        /*  _i2c_drv[3] = 0x00;
            i2c_drv[4] = 0x00;     */

            _led_iic_tx(0x0F, 23, _i2c_drv);
            _i2c_3ms_count = 0;
        }
    }
```

This functions will runs in every 10 msec.
It sends the 23 bytes I2C data to drive the LED.

OUT0_COLOR	0Fh	R/$\overline{W}$	OUT0_Color	00h
OUT1_COLOR	10h	R/$\overline{W}$	OUT1_Color	00h
OUT2_COLOR	11h	R/$\overline{W}$	OUT2_Color	00h
OUT3_COLOR	12h	R/$\overline{W}$	OUT3_Color	00h