

# LP5569 Lighting Pattern Design

# ABSTRACT

This application report describes how to design lighting pattern with LP5569 Programmable lighting engine and provides examples of the fashion lighting pattern for the user.

1	Introduction	1
2	Device Overview	2
3	Lighting Pattern Design with LEDs on LP5569EVM	2
4	Lighting Pattern Design with LEDs on LP5569 Ring Demo	5
	References	

### List of Figures

LED Board Design	5
Mono-Color Chasing Engine Code for 4 Devices	7
Multi-Color Chasing Code for U1 and U2	8
Multi-Color Chasing Code for U3 and U5	9
Door Open	10
lasm.exe Path	11
Compile Panel in LP5569EVM GUI	12
	Mono-Color Chasing Engine Code for 4 Devices Multi-Color Chasing Code for U1 and U2 Multi-Color Chasing Code for U3 and U5 Door Open lasm.exe Path

### List of Tables

1	LED and I2C Address Assignment	5
•		× .

# Trademarks

All trademarks are the property of their respective owners.

## 1 Introduction

This application report describes LP5559 lighting pattern design with examples for the device quick start. Most of the programs are presented with command compiler syntax. The Command compiler is described in The Control View - Source Edit Tab of Using the BOOST-LP5569EVM Evaluation Module. This application report also includes LED ring Demo (click here) sample code in both command syntax and C language. Command compiler software is available with the evaluation kit.



# 2 Device Overview

The LP5569 device is a programmable, easy-to-use 9-channel I2C LED driver designed to produce lighting effects for various applications. The LED driver is equipped with an internal SRAM memory for user programmed sequences and three programmable LED engines, which allow operation without processor control. Autonomous operation reduces system power consumption when the processor is put in sleep mode.

# 3 Lighting Pattern Design with LEDs on LP5569EVM

The following lighting pattern is realized with white LEDs on LP5569EVM.

## 3.1 Bouncing Effect

This design is best viewed with 9 white LEDs on the LP5569EVM. This design has 4 white LEDs on at all times. Each LED has different PWM settings to create two tracers bouncing back and forth.

; bouncing.src

L10:	dw	000000	00000000001	C		
L11:	dw	000000	00000000010k	С		
L12:	dw	000000	00000000100k	C		
L13:	dw	000000	00000001000k	С		
L14:	dw	000000	00000010000k	C		
L15:	dw	000000	00000001000k	C		
L16:	dw	000000	00000000100k	C		
L17:	dw	000000	000000000101	C		
L20:	dw	000000	00100000000	C		
L21:	dw	000000	00010000000	C		
L22:	dw	000000	0000100000b	C		
L23:	dw	000000	00000100000	C		
L24:	dw	000000	0000010000	C		
L25:	dw	000000	00000100001	C		
L26:	dw	000000	0000100000b	C		
L27:	dw	000000	00010000000	C		
.segmen	t pro	ogram1				
	map_sta	art	L10	;load	the start address	
	load_er	nd	L17	;load	the end address	
loop1:						
	triggen	<u>c</u>	s{2}			
	set_pwr	n	0			
	map_nex	ĸt				
	set_pwr	n	30			
	map_nex	ĸt				
	set_pwr	n	100			
	map_pre	≥v				
	wait		0.1	;wait	time to create effect	
	branch		0, loopl			
	end					
.segmen	t pro	ogram2				
	map_sta	art	L20	;load	the start address	
	load_er	nd	L27	;load	the end address	
loop2:						
	triggen	<u>_</u>	w{1}			
	set_pwr	n	0			
	map_nex	ĸt				
	set_pwr	n	30			
	map_nex	ĸt				
	set_pwr	n	100			
	map_pre	ev				
	branch		0, loop2			
	end					
.segment	t		program3			
	end					



# 3.2 Breath\_white Effect

This design is best viewed with 9 white LEDs on the LP5569EVM. This design has 3 group LED breathing.

```
GRP1:
                     000000001001001b
           dw
GRP2:
           dw
                     000000010010010b
GRP3:
           dw
                     0000000100100100b
.segment
        programl
                         ;Begin of a segment
       map_addr GRP1
       set_pwm
                  00
loop1:
               1, 100
1, -100
       ramp
       ramp
       wait
                0.3
       branch
                0, loopl
       end
.segment program2
                         ;Begin of a segment
       map_addr GRP2
                  00
       set_pwm
loop2:
                 1, 100
       ramp
       ramp
                1, -100
       wait
                 0.3
                0, loop2
       branch
       end
.segment program3
                         ;Begin of a segment
       map_addr GRP3
       set_pwm
                  00
loop3:
                1, 100
       ramp
       ramp
                1, -100
       wait
                 0.3
       branch
                 0, loop3
       end
```



# 3.3 Chaser Effect

This design is best viewed with 9 white LEDs on the LP5569EVM. This design has 3 white LEDs on at all times. Each LED has different PWM settings to create a chaser (or tracer) effect.

	er.src	
;		
L00:	dw	0000000000000001b
L01:	dw	0000000000000010b
L02:	dw	0000000000000100b
L03:	dw	0000000000001000b
L04:	dw	000000000010000b
L05:	dw	00000000010000b
L06:	dw	00000000100000b
L07:	dw	00000001000000b
L08:	dw	00000010000000b
L09:	dw	00000001000000b
L10:	dw	00000000100000b
L11:	dw	00000000010000b
L12:	dw	000000000010000b
L13:	dw	0000000000001000b
L14:	dw	0000000000000100b
L15:	dw	0000000000000010b
ALL:	dw	000000011111111b

.segmen	t program	L	
	map_addr	ALL	
	ramp	0.5, 150	
	ramp	0.5, -150	
	wait	0.2	
	map_start	L00	;load the start address
	load_end	L15	;load the end address
loop1:			
	set_pwm	0	
	map_next		
	set_pwm	5	
	map_next		
	set_pwm	50	
	map_next		
	set_pwm	150	
	map_prev		
	map_prev		
	wait	0.07	;wait time to create effect
	branch	0, loopl	
	end		
.segmen	t program	2	

end



# 4 Lighting Pattern Design with LEDs on LP5569 Ring Demo

The LED ring is the latest HMI in the smart personal electronic device space and improves the user experience. More and more appliance vendors are adopting this concept in next generation products. For the appliance customer, the existing model with the existing MCU already meets system specification, but the fancy lighting pattern will exhaust the system resources and potentially cause the MCU to crash. It is requires a triple design circle to design complex lighting patterns without an engine control LED driver.

According to Figure 1 for LED board design, U1,U2,U3, and U5 drive 12 pcs RGB LED modules. The LED mapping and I2C address assignments are shown as the Table 1.

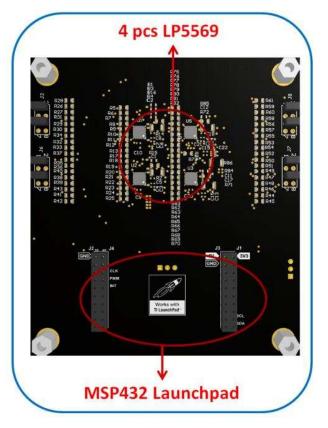




Figure 1. LED Board Design

# Table 1. LED and I2C Address Assignment

Device	I2C Address	Broadcasting I2C Address	Channel Number of The LED Driver	LED
U1	0x32H	0x40H	LED0, LED3, LED6	D1-B, D1-G,D1-R
U1	0x32H	0x40H	LED1, LED4, LED7	D2-B, D2-G,D2-R
U1	0x32H	0x40H	LED2,LED5, LED8	D3-B, D3-G,D3-R
U2	0x33H	0x40H	LED0, LED3, LED6	D4-B, D4-G,D4-R
U2	0x33H	0x40H	LED1, LED4, LED7	D5-B, D5-G,D5-R
U2	0x33H	0x40H	LED2,LED5, LED8	D6-B, D6-G,D6-R
U3	0x34H	0x40H	LED0, LED3, LED6	D7-B, D7-G,D7-R
U3	0x34H	0x40H	LED1, LED4, LED7	D8-B, D8-G,D8-R
U3	0x34H	0x40H	LED2,LED5, LED8	D9-B, D9-G,D9-R
U5	0x35H	0x40H	LED0, LED3, LED6	D10-B, D10-G,D10-R
U5	0x35H	0x40H	LED1, LED4, LED7	D11-B, D11-G,D11-R
U5	0x35H	0x40H	LED2,LED5, LED8	D12-B, D12-G,D12-R



### Lighting Pattern Design with LEDs on LP5569 Ring Demo

Firstly, define the LED Mapping in the beginning of the engine coding as shown below.

-					0	•		
row1:	dw	0000000001001001b	;Map	B LED =	D1, D4,	D7 on	the eval.	board.
	dw	0000000010010010b	;Map	G LED =	D2, D5,	D8 on	the eval.	board.
	dw	000000100100100b	;Map	R LED =	D3, D6,	D9 on	the eval.	board.
	dw	0000000011011011b	;Map	BG LED c	on the e	val. bo	bard.	
	dw	0000000110110110b	;Map	GR LED c	on the e	val. bo	bard.	
	dw	0000000101101101b	;Map	RB LED c	on the e	val. bo	bard.	
row7:	dw	0000000111111111b	;Map	all LEDs	s on the	eval.	board.	
row8:	dw	0000000001001001b	;Map	B LED =	D1,D4,D	7 on th	ne eval. k	board.
row9:	dw	0000000010010010b	;Map	G LED =	D2,D5,D	8 on th	ne eval. k	board.

# 4.1 Breathing

During the breathing pattern, all LEDs fade in and out as the same color at the same rate, therefore all devices should run the same engine code below.

.segmen loop1 0		raml ;Pr	ogram for engine 1.
10001_0	map_start load_end		;Map the first LED. ;End address of the mapping data table.
loop1:			
	ramp	2, 200	;Increase PWM 0->78% in 2 second.
	ramp	2, -255	;Decrease PWM ->0% in 2 seconds.
	wait	0.4	;Wait for 0.4 seconds.
	ramp	2, 200	;Increase PWM 0->78% in 2 second.
	ramp	2, -255	;Decrease PWM ->0% in 2 seconds.
	wait	0.4	;Wait for 0.4 seconds.
	map_next		;Set the next row active in the mapping table.
	branch	6,loopl	;Loop 6 time
	map_addr		
	ramp	1.5, 200	;Increase PWM 0->78% in 1.5 second.
	wait	0.4	;Wait for 0.4 seconds.
	wait		;Wait for0.4 seconds.
	map_addr		
	ramp	3, 200	;Increase PWM 0->78% in 3 second.
	ramp	3, -255	
	ramp	3, 200	
	ramp		;Decrease PWM ->0% in 3 seconds.
	map_addr		
	ramp	1.5, -255	;Decrease PWM ->0% in 1.5 seconds.



Lighting Pattern Design with LEDs on LP5569 Ring Demo

### www.ti.com

# 4.2 Mono-Color Chasing

The mono-color chasing pattern needs the devices to start execution with a specific sequence delay. Each device has different code, as shown in Figure 2.

	wait	0.4			wait	0.4
	ld	ra, 75			ld	ra, 135
loop1_2:	map_sel	7		loop1_2:	map sel	7
	set_pwm	ra		. –	set_pwm	ra
	map_sel	4			map_sel	4
	sub	ra, 20			sub	ra, 20
	set_pwm	ra			set_pwm	ra
	map_sel	1			map_sel	1
	sub	ra, 20			sub	ra, 20
	set_pwm	ra			set_pwm	ra
	add	ra, 20			add	ra, 20
	wait	0.03			wait	0.03
	branch	63, loop1_2			branch	63, loop1_2
	branch	10, loop1_2			branch	3, loop1_2
	wait	0.1			wait	0.1
	map_addr	row7			map_addr	row7
	set_pwm	0	U1	U2	set_pwm	0
	wait	0.4			wait	0.4
	wait Id	0.4 ra. 255	U5	U3	wait Id	0.4 ra. 195
loop1 2:	ld	ra, 255	U5		ld	ra, 195
loop1_2:	ld map_sel		U5	U3 loop1_2:	ld map_sel	
loop1_2:	ld map_sel set_pwm	ra, 255 7 ra	U5		ld map_sel set_pwm	ra, 195 7
loop1_2:	ld map_sel	ra, 255 7	U5		ld map_sel	ra, 195 7 ra 4
loop1_2:	ld map_sel set_pwm map_sel sub	ra, 255 7 ra 4	U5		ld map_sel set_pwm map_sel sub	ra, 195 7 ra
loop1_2:	ld map_sel set_pwm map_sel	ra, 255 7 ra 4 ra, 20	U5		ld map_sel set_pwm map_sel	ra, 195 7 ra 4 ra, 20
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm	ra, 255 7 ra 4 ra, 20 ra	U5		ld map_sel set_pwm map_sel sub set_pwm	ra, 195 7 ra 4 ra, 20 ra
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm map_sel	ra, 255 7 ra 4 ra, 20 ra 1	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel	ra, 195 7 ra 4 ra, 20 ra 1
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm map_sel sub	ra, 255 7 ra 4 ra, 20 ra 1 ra, 20	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel sub	ra, 195 7 ra 4 ra, 20 ra 1 ra, 20
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm	ra, 255 7 ra 4 ra, 20 ra 1 ra, 20 ra	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm	ra, 195 7 ra 4 ra, 20 ra 1 ra, 20 ra
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm add	ra, 255 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm add	ra, 195 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm add wait	ra, 255 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm add wait	ra, 195 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm add set_pwm add wait branch branch wait	ra, 255 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03 63, loop1_2	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm add wait branch	ra, 195 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03 63, loop1_2
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm add set_pwm add wait branch branch	ra, 255 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03 63, loop1_2 3, loop1_2	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm add wait branch branch	ra, 195 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03 63, loop1_2 3, loop1_2
loop1_2:	ld map_sel set_pwm map_sel sub set_pwm add set_pwm add wait branch branch wait	ra, 255 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03 63, loop1_2 3, loop1_2 0.1	U5		ld map_sel set_pwm map_sel sub set_pwm map_sel sub set_pwm add wait branch branch wait	ra, 195 7 ra 4 ra, 20 ra 1 ra, 20 ra ra, 20 0.03 63, loop1_2 3, loop1_2 0.1

Figure 2. Mono-Color Chasing Engine Code for 4 Devices



# 4.3 Multi-Color Chasing

The multi-color chasing pattern requires the devices to start execution with a specific sequence delay. Each device has different code, as shown in Figure 3 and Figure 4.

.segment	program2						
loop2_3:	wait	0.08	U1	U2			
	branch	2, loop2_3	01	02			
	map_clr			.segment	program2		
	trigger	s{3}			trigger	w{1}	
	rst				map_addr	row8	
	trigger	w{1}			ramp	1, 200	
	map_addr	row8		loop2_1:	wait	0.08	
	ramp	1, 200			branch	3, loop2_1	
loop2_1:	wait	0.08		loop2_2:	map_sel	2	
loop2_2:	map_sel	2			set_pwm	200	
	set_pwm	200			wait	0.08	
	wait	0.08			map_sel	5	
	map_sel	5			set_pwm	200	
	set_pwm	200			wait	0.08	
	wait	0.08			map_sel	8	
	map_sel	8			set_pwm	200	
	set_pwm	200 0.08			wait	0.08	
	wait				map_sel	2 0	
	map_sel	2 0			set_pwm	0.08	
	set_pwm wait	0.08			wait map sel	5	
	map_sel	5			set_pwm	0	
	set pwm	0			wait	0.08	
	wait	0.08			map sel	8	
	map_sel	8			set_pwm	0	
	set pwm	0			wait	0.08	
	wait	0.08			map_sel	3	
	map sel	3			set pwm	200	
	set_pwm	200			wait	0.08	
	wait	0.08			map_sel	6	
	map_sel	6			set pwm	200	
	set_pwm	200			wait	0.08	
	wait	0.08			map_sel	9	
	map_sel	9			set_pwm	200	
	set_pwm	200			wait	0.08	
	wait	0.08			map_sel	3	
	map_sel	3			set_pwm	0	
	set_pwm	0			wait	0.08	
	wait	0.08			map_sel	6	
	map_sel	6			set_pwm	0	
	set_pwm	0			wait	0.08	
	wait	0.08			map_sel	9	
	map_sel	9			set_pwm	0	
	set_pwm	0			wait	0.08	
	wait	0.08			branch	63, loop2_2	
	branch	63, loop2_2			map_addr	row7 0	
	map_addr	row7			set_pwm	0	
Jacob 2, 20	set_pwm	0 0.08			map_clr	e(2)	
loop2_3: v	branch				trigger	s{3}	
	map clr	2, loop2_3			rst		
	trigger	s{3}					
	rst	2(2)					
	150						

Figure 3. Multi-Color Chasing Code for U1 and U2



# Lighting Pattern Design with LEDs on LP5569 Ring Demo

ogmont	program?				.segment	
segment	program2 trigger	w{1}	U5			trigger
	map_addr	row8	05			map_addr
	ramp	1, 200				ramp
oop2_1:	wait	0.08			oop2_1:	
100p2_1.	branch	3, loop2_1		loop2_2	:	: map_sel
loop2_2:	map sel	3				set_pwm
100p2_2.	set_pwm	200			wait	
	wait	0.08			map_se	el
	map_sel	6			set_pwm	
	set pwm	200			wait	
	wait	0.08			map_sel	
	map_sel	9			set_pwm	
	set_pwm	200			wait	
	wait	0.08			map_sel	
	map_sel	3			set_pwm	
	set_pwm	0			wait	
	wait	0.08			map_sel	
	map_sel	6			set_pwm	
	set_pwm	0			wait	
	wait	0.08			map_sel	
	map_sel	9			set_pwm	
	set_pwm	0			wait	
	wait	0.08			map_sel	
	map sel	2			set_pwm	
	set_pwm	200			wait	
	wait	0.08			map_sel	
	map_sel	5			set_pwm	
	set_pwm	200			wait	
	wait	0.08			map_sel	
	map_sel	8			set_pwm	
	set_pwm	200			wait	
	wait	0.08			map_sel	
	map_sel	2			set_pwm	
	set pwm	0			wait	
	wait	0.08			map_sel	
	map_sel	5			set_pwm	
	set pwm	0			wait	
	wait	0.08			map_sel	
	map_sel	8			set_pwm	
	set pwm	0			wait	
	wait	0.08			branch	
	branch	63, loop2_2			map_addr	
	map addr	row7			set_pwm	
	set pwm	0		loop2_3:	wait	
	map_clr	0			branch	
	trigger	s{3}			map_clr	
	rst	3(3)			trigger	
	150				rst	

Figure 4. Multi-Color Chasing Code for U3 and U5



# 4.4 Door Open

The door open pattern needs the devices to start execution with a specific sequence delay. Each device has different code as shown in Figure 5.

1	- 11	0.4		1	- 9	ô. 1
loop1_8:	wait	0.4		loop1_8:	wait	0.4
	map_sel	1		loop1_9:	wait	0.05
	set_pwm wait	200 0.05			branch	2, loop1_9 1
	map_sel	4			map_sel set pwm	200
	set_pwm	200			wait	0.05
	wait	0.05			map_sel	4
	map_sel	7			set_pwm	200
	set_pwm	200			wait	0.05
loop1_9:	wait	0.05			map_sel	7
	branch	6, loop1_9			set_pwm	200
	wait	0.4			wait	0.05
	map_sel	7			wait	0.4
	set_pwm	0			map_sel	7
	wait	0.05			set_pwm	0
	map_sel	4			wait	0.05
	set_pwm	0			map_sel	4
	wait	0.05			set_pwm	0
	map_sel	1			wait	0.05
	set_pwm	0			map_sel	1
	wait	0.05			set_pwm	0
	branch	63, loop1_8		loop1_10:		0.05
	map_clr				branch	3, loop1_10
	trigger	s{2}			branch	63, loop1_8
	trigger	w{3}	U1	U2	map_clr	
	rst		01	02	trigger	s{2}
					trigger	w{3}
					rst	
loop1_8:	wait	0.4		loop1_8:	wait	0.4
loop1_8: loop1_9:	wait	0.05			wait map_sel	7
	wait branch	0.05 2, loop1_9	U5	loop1_8: U3	map_sel set_pwm	7 200
	wait branch map_sel	0.05 2, loop1_9 7	U5		map_sel set_pwm wait	7 200 0.05
	wait branch map_sel set_pwm	0.05 2, loop1_9 7 200	U5		map_sel set_pwm wait map_sel	7 200 0.05 4
	wait branch map_sel set_pwm wait	0.05 2, loop1_9 7 200 0.05	U5		map_sel set_pwm wait map_sel set_pwm	7 200 0.05 4 200
	wait branch map_sel set_pwm wait map_sel	0.05 2,loop1_9 7 200 0.05 4	U5		map_sel set_pwm wait map_sel set_pwm wait	7 200 0.05 4 200 0.05
	wait branch map_sel set_pwm wait map_sel set_pwm	0.05 2,loop1_9 7 200 0.05 4 200	U5		map_sel set_pwm wait map_sel set_pwm wait map_sel	7 200 0.05 4 200 0.05 1
	wait branch map_sel set_pwm wait map_sel set_pwm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm	7 200 0.05 4 200 0.05 1 200
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1	U5		map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait	7 200 0.05 4 200 0.05 1 200 0.05
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.05 0.4	U5	U3	map_sel set_pwm wait map_sel set_pwm wait set_pwm wait branch wait map_sel	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait wait map_sel	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait wait map_sel set_pwm	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.4 1 0 0.05	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait map_sel	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0 0.05 4
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait set_pwm wait set_pwm wait set_pwm wait	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0 0.05 4
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0.05	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait map_sel set_pwm	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0.05 4
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0 0.05 4 0 0.05	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait map_sel set_pwm wait map_sel	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0 0.05 7
	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0.05 4 0 0.05 7	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait set_pwm wait set_pwm wait set_pwm wait set_pwm wait set_pwm wait set_pwm	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0 0 0.05 7
loop1_9:	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0 0.05 4 0 0.05 7	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait set_pwm wait set_pwm wait set_pwm wait set_pwm wait set_pwm wait	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0 0 0.05 7 0 0.05 7
loop1_9:	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait set_pwm wait set_pwm wait set_pm wait set_pm wait set_pm wait set_pm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0 0.05 4 0 0.05 7 0 0.05 7	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0 0.05 7 0 0.05 7
loop1_9:	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0 0.05 4 0 0.05 7 0 0.05 7	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0 0.05 7 0 0.05 7 0 0.05 5 7 0 0.05 6, loop1_8 0.05 7 0 0.05 6, loop1_8 0.05 7 0 0.05 6, loop1_8 0.05 7 0 0.05 7 0 0.05 7 0 0.05 7 0 0.05 7 0 0.05 7 0 0.05 7 0 0.05 0 0 0 0 0 0 0 0 0 0 0 0 0
loop1_9:	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait set_pwm wait set_pwm wait set_pm wait set_pm wait set_pm wait set_pm wait	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0 0.05 4 0 0.05 7 0 0.05 7	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch map_clr trigger	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0 0.05 7 0 0.05 7 0 0.05 5, loop1_8 s{2}
loop1_9:	wait branch map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch branch map_cl	0.05 2, loop1_9 7 200 0.05 4 200 0.05 1 200 0.05 0.4 1 0 0.05 4 0 0.05 7 0 0.05 7 0 0.05 3, loop1_10 63, loop1_8	U5	U3	map_sel set_pwm wait map_sel set_pwm wait map_sel set_pwm wait branch wait map_sel set_pwm wait set_pwm wait set_pwm wait set_pwm wait set_pwm wait set_pwm wait branch map_sel set_pwm wait branch map_sel set_pwm	7 200 0.05 4 200 0.05 1 200 0.05 6, loop1_9 0.4 1 0 0.05 4 0 0.05 7 0 0.05 7 0 0.05 5, loop1_8 s{2}

Figure 5. Door Open

SNVA822–May 2018 Submit Documentation Feedback



# 4.5 Coding Tips

- Use the branch instruction to synchronize multiple devices, as it guarantees all devices will be at the same time scale and step.
- Remember to clear the LED mapping with map\_clr when using the same LED in the difference engine.

Lighting Pattern Design with LEDs on LP5569 Ring Demo

• Use the appropriate variable (ra, rb, rc, rd) for the progressive increase or decrease operation.

# 4.6 Uploading The Program to SRAM

The compile tool (Lasm.exe) can be downloaded from the GUI package on ti.com. The command window or GUI can be used to compile the .scr file.

• New folder			
Name A	Date modified	Туре	Size
👃 driverlib	5/10/2017 5:24 PM	File folder	
🐌 DSLite	4/8/2018 10:54 AM	File folder	
🗼 Info	4/8/2018 10:54 AM	File folder	
L rollbackBackupDirectory	4/24/2017 2:11 PM	File folder	
📙 rollbackBackupDirectory1	5/10/2017 5:06 PM	File folder	
lockBackupDirectory2	5/25/2017 12:15 PM	File folder	
👃 rollbackBackupDirectory3	5/26/2017 10:34 AM	File folder	
🐌 rollbackBackupDirectory4	4/8/2018 10:54 AM	File folder	
📜 rom	5/10/2017 5:24 PM	File folder	
📙 targetConfigs	5/10/2017 5:24 PM	File folder	
.ccsproject	5/10/2017 5:21 PM	CCSPROJECT File	1 KB
cproject	5/10/2017 5:21 PM	CPROJECT File	27 KB
.project	5/10/2017 5:21 PM	PROJECT File	1 KB
🔲 🔳 data.dat	5/10/2017 5:31 AM	DAT File	70 KB
🚰 info.htm	5/10/2017 5:31 AM	HTML Document	3 KB
🔳 lasm.exe	5/10/2017 5:31 AM	Application	58 KB
E LP5569_EVM.exe	7/5/2017 9:15 AM	Application	879 KB
	E 113 13017 1 00 DN1		24 1/0

Figure 6. lasm.exe Path



1U								
12C 0x32 400kHz	•	Direct Access Reg(hex) Data(hex) mory Source Ed	0x000 0x0000	Read Write	07 06 05	04 03 02 01 00	E	ENVPWM EN PWM 20000 v Hz 50 v % Update 227KHz Rest
New	Load	Save	Save As	Comment		Save and Compile	] 	
0 1 2 3 4 5								Comment Line Clear Line Remove Line Insert Line Replace
6 7 8 9							ш	E Compile Log
10 11 12								Comple Eug

Figure 7. Compile Panel in LP5569EVM GUI

After the compiling, a .hex file will appear in the same folder as the .scr file.



Then, copy the hex file to the array table and upload the data to the SRAM by the below coding.

```
void load_SRAM()
{
   int i.i;
   MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, 0x40);
                                                                    //Device global setting
       MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x2F);
                                                                    //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x48);
                                                                    //send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x02);
                                                                    //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x54);
                                                                    //send register data
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, 0x32);
                                                                    //load hex to SRAM in U1
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4b);
                                                                    //send register address
                                                                    //send register data
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x09);
       MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4c);
                                                                   //send register address
       MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x78);
                                                                   //send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4d);
                                                                    //send register address
       MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE, 0xa8);
                                                                    //send register data
        for(j=0; j<16; j++)</pre>
        ł
            MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4F); //send register address
            MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,j);
                                                                   //send register data
            MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE,0x50); //send register address
            for(i=0; i<32; i++)</pre>
             MAP_I2C_masterSendMultiByteNext(EUSCI_B1_BASE,table_32[i+j*32]);//send register data
        }
   MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, 0x33);
                                                                    //load hex to SRAM in U2
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4b);
                                                                    //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x09);
                                                                    //send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4c);
                                                                    //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x7b);
                                                                   //send register data
       MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4d);
                                                                   //send register address
       MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE, 0xaa);
                                                                    //send register data
        for(j=0; j<16; j++)</pre>
        {
            MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4F); //send register address
            MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,j);
                                                                   //send register data
            MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE,0x50); //send register address
            for(i=0; i<32; i++)</pre>
             MAP_I2C_masterSendMultiByteNext(EUSCI_B1_BASE,table_33[i+j*32]);//send register data
        }
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, 0x34);
                                                                    //load hex to SRAM in U3
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4b);
                                                                   //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x09);
                                                                   //send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4c);
                                                                   //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x7b);
                                                                   //send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4d);
                                                                   //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0xab);
                                                                    //send register data
        for(j=0; j<16; j++)</pre>
        {
            MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4F); //send register address
            MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,j); //send register data
            MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE,0x50); //send register address
            for(i=0; i<32; i++)
             MAP_I2C_masterSendMultiByteNext(EUSCI_B1_BASE,table_34[i+j*32]);//send register data
        }
                                                                    //load hex to SRAM in U5
   MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, 0x35);
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4b);
                                                                    //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE, 0x09);
                                                                    //send register data
       MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4c);
                                                                    //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x77);
                                                                    //send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4d);
                                                                    //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE, 0xa6);
                                                                    //send register data
```

ences		www.ti.com
	for(j=0; j<16; j++) {	
	<pre>MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x4F); MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,j); MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,j);</pre>	//send register data
	<pre>MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE,0x50); for(i=0; i&lt;32; i++)</pre>	//send register address
	<pre>MAP_I2C_masterSendMultiByteNext(EUSCI_B1_BASE,table_3) }</pre>	5[i+j*32]);//send register data;
MZ	AP_I2C_setSlaveAddress(EUSCI_B1_BASE, 0x32);	//SET U1 as the clk out
	<pre>MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x2F);</pre>	//send register address
	MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x49);	//send register data
	MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x3d);	//send register address
	<pre>MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x08);</pre>	//send register data
MAP_I2	<pre>2C_setSlaveAddress(EUSCI_B1_BASE, 0x40);</pre>	//Run all the engine
MZ	AP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x02);	//send register address
MZ	AP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x00);	//send register data
MZ	AP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x02);	//send register address
MZ	AP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0xa8);	//send register data
MZ	AP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x01);	//send register address
MZ	AP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0xa8);	//send register data

```
}
```

# 5 References

Using the BOOST-LP5569EVM Evaluation Module LP5569 Nine-Channel I2C RGB LED Driver With Engine Control and Charge Pump TEXAS INSTRUMENTS

### IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your noncompliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products <a href="http://www.ti.com/sc/docs/stdterms.htm">http://www.ti.com/sc/docs/stdterms.htm</a>), evaluation modules, and samples (<a href="http://www.ti.com/sc/docs/stdterms.htm">http://www.ti.com/sc/docs/stdterms.htm</a>), evaluation

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2018, Texas Instruments Incorporated