SPECIFICATIONS FOR
LIQUID CRYSTAL DISPLAY MODULE

MODEL NO : ALP100010-60RGB-UR

CUSTOMER :

APPROVED SIGNATURE

DSGD :

CHKD : Peter

APPD : Peng Jun

DATE : 2019-03-20

YUDU AMSON ELECTRONICS CO.,LTD.

YuDu Industrial Garden,Ganzhou City,JiangXi,China

TEL : 86-797-6330063
FAX : 86-797-6330055

Revision Record

No.	Date	Model No.	Version	Remarks
1	2019-03-20	ALP100010-60RGB-UR	REV.0	Spec RoHs-Compliant

MECHANICAL SPECIFICATION

ITEM	DESCRIPTION
Product No.	ALP100010-60RGB-UR
Controller Board Size	40.0(W)×20.0(H)×8.1max(D) mm
Light Bar Size	1004.0(W)×10.0(H)×3.0(D) mm
Controller Interface	STM32F030C8T6 UART
ROM Selection	-
Built-in	With DC/DC Converter
Module Weight	T.B.D

PIN ASSIGNMENTS

CN1

Pin No.	Pin Out	Level	Description
1.	**TX1**	H/L	Serial Transmit Signal
2.	**RX1**	H/L	Serial Receive Signal
3.	**+5V**	+5V	Power Supply Voltage
4.	**VSS**	0V	Power Supply Ground

USB

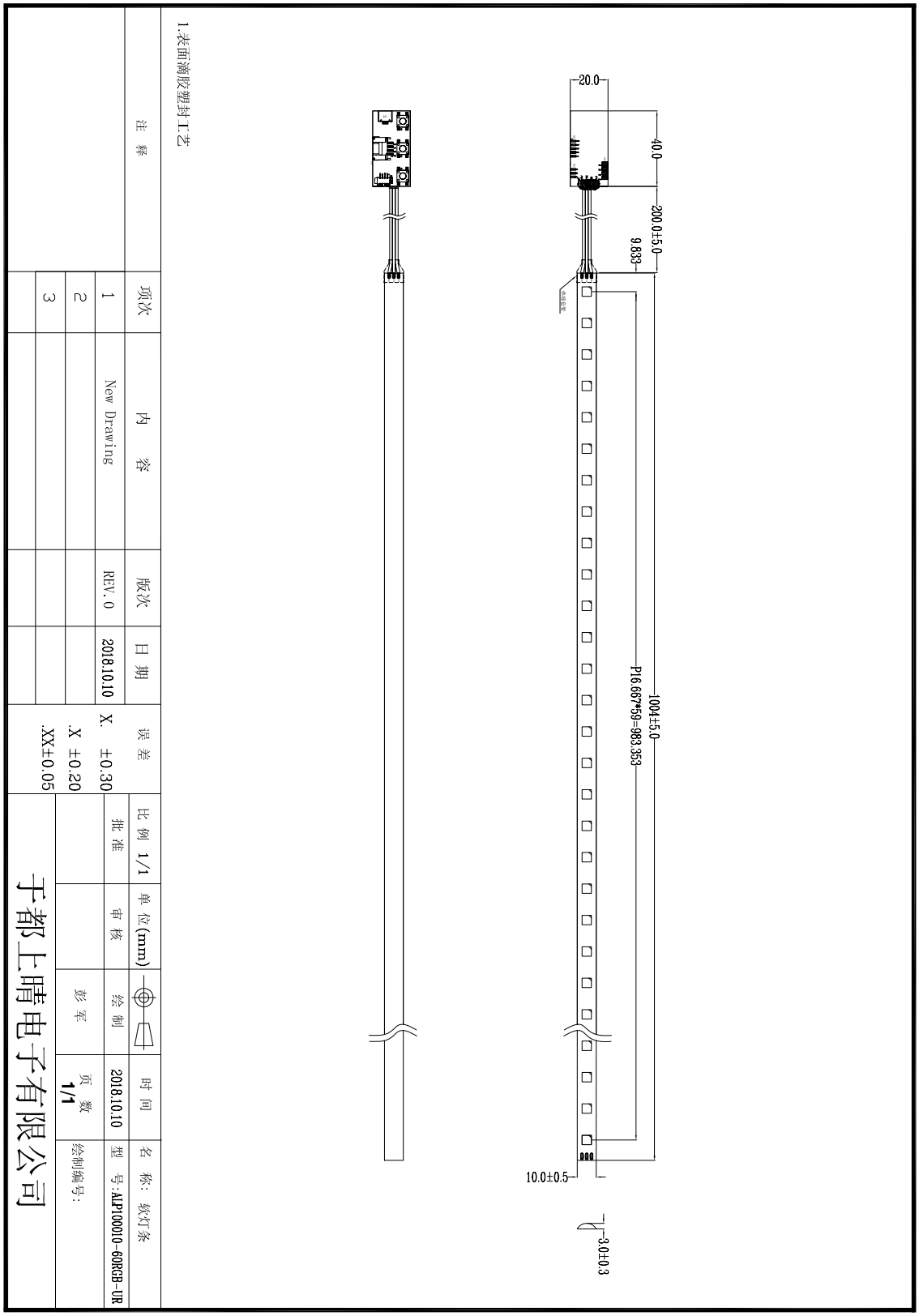
Pin No.	Pin Out	Level	Description
1.	**+5V**	+5V	Power Supply Voltage
2.	**NC**		No connect
3.	**NC**		No connect
4.	**NC**		No connect
5.	**VSS**	0V	Power Supply Ground

Key

Pin No.	Description
S1	Set the dynamic function
S2	Set dynamic fucntion's speed
S3	Set the dynamic fucntion's color

ELECTRO-OPTICAL CHARACTERISTIC TA=25℃

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
VDD	Power Supply			5.0	5.5	V
VIH	Input High Voltage		0.7VDD		VDD+0.3	
VIL	Input Low Voltage		VSS -0.3		0.3VDD	
IDD	Power Supply current	@VDD=5V	70	-	2230	Ma
	PEAK WAVELENGTH (PER DOT)	R	620		625	NM
		G	522		525	
		B	467.5		470	
	LUMINOUS INTENSITY (PER DOT)	R	360		500	MCD
		G	1100		1800	
		B	200		800	
FOSC	Oscillator Frequency			800		KHz


20.0
40.0
200.0±5.0
9.833
P16.667*59=983.353
1004±5.0
10.0±0.5
3.0±0.3
1.表面滴胶塑封工艺
注 释
项次
内 容
版次
日 期
1
New Drawing
REV.0
2018.10.10
2
3
误 差
X. ±0.30
.X ±0.20
.XX±0.05
比 例 1/1
单位(mm)
批 准
审 核
彭 军
绘 制
时 间
2018.10.10
页 数 1/1
于都上晴电子有限公司
名 称: 软灯条
型 号:ALP100010-60RGB-UR
绘制编号:

ezDisplay RGB Ring and Stripe Command List

Code	Function	Digital RGB LED Ring and Stripe has 256 grayscale for each primary	
		Instruction of AT Command mode	**API for C code**
0xc0	Set the color of desinated pixel	1. atc0=(address of pixel, grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc0=(0,255,255,0)	printf("atc0=(%d,%d,%d,%d)",address,R,G,B); while (USART_ReceiveData(UART1)!='E') {}
0xc1	Set the color of desinated pixels within a section	1. atc1=(address of the start pixel, address of the end pixel, grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc1=(18,25,0,100,0)	printf("atc1=(%d,%d,%d,%d,%d)",address1,address2,R,G,B); while (USART_ReceiveData(UART1)!='E') {}
0xc2	Set the color randomly for each pixel of ring	1. atc2=() 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc2=()	printf("atc2=()"); while (USART_ReceiveData(UART1)!='E') {}
0xc3	Turn the ring pixels clockwise one round	1. atc3=(speed of turning 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc3=(10)	printf("atc3=(%d)",speed); while (USART_ReceiveData(UART1)!='E') {}
0xc4	Turn the ring pixels counter clockwise one round	1. atc4=(speed of turning 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc4=(10)	printf("atc4=(%d)",speed); while (USART_ReceiveData(UART1)!='E') {}
0xc5	Turn one pixels Clockwise	1. atc5=(speed of shifting 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atcb=(10)	printf("atc5=(%d)",speed); while (USART_ReceiveData(UART1)!='E') {}
0xc6	Turn one pixels Counter clockwise	1. atc6=(speed of shifting 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atcc=(10)	printf("atc6=(%d)",speed); while (USART_ReceiveData(UART1)!='E') {}
0xc7	Flash one desinated pixle	1. atc7=(address of pixel, speed of flashing 1~100) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc7=(0,50)	printf("atc7=(%d,%d)",address,speed); while (USART_ReceiveData(UART1)!='E') {}
0xc8	Flash desinated pixels within a section	1. atc8=(address of the start pixel, address of the end pixel, speed of flashing 1~100) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc8=(2,5,50)	printf("atc8=(%d,%d,%d)",address1,address2,speed); while (USART_ReceiveData(UART1)!='E') {}
0xc9	Flash whole ring	1. atc9=(speed of flashing 1~100) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atc9=(50)	printf("atc9=(%d)",speed); while (USART_ReceiveData(UART1)!='E') {}
0xca	Breath effect of whole ring for 7 major colors	1. atca=(0 or 1 for R, 0 or1 for G, 0 or 1 for B) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atca=(0,0,1)	printf("atca=(%d,%d,%d)",R,G,B); while (USART_ReceiveData(UART1)!='E') {}
0xcd	Set the dynamic fucntion's color	1. atcd=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atcd=(128,9,18)	printf("atcd=(%d,%d,%d)",R,G,B); while (USART_ReceiveData(UART1)!='E') {}

0xce	Set the dynamic fucntion's speed	1. atce=(speed 1~100) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atce=(5)	printf("atce=(%d)",speed); while (USART_ReceiveData(UART1)!= 'E') {}
0xcf	Set the pixel number of ring	1. atcf=(number of pixels of ring 1~120) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atcf=(48)	printf("atcc=(%d)",Number_of_Pixel); while (USART_ReceiveData(UART1)!= 'E') {}
0xd0	Clear display	1. atd0=() 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atd0=()	printf("atd0=()"); while (USART_ReceiveData(UART1)!= 'E') {}
0x10	Fill pixel one by one, strat from last pixel	1. at10=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of filling 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at10=(0,105,0,5)	printf("at10=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0x11	Fill pixel one by one, strat from first pixel	1. at11=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of filling 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at11=(0,105,0,2)	printf("at11=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0x12	Stack pixel one by one clockwise then turn off pixel counterclockwise	1. at12=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of stacking 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at12=(0,10,255,5)	printf("at12=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0x13	Stack pixel one by one counterclockwise then turn off pixel clockwise	1. at13=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of stacking 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at13=(255,0,0,10)	printf("at13=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0x14	Two pixels collision then firework	1. at14=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of stacking 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at14=(0,255,0,10)	printf("at14=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0x15	Two stack pixels collision then firework	1. at15=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of stacking 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at15=(0,255,0,10)	printf("at15=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0x16	Two pixels collision then bounce back	1. at16=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of stacking 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at16=(255,255,255,10)	printf("at16=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0x17	Two stack pixels collision then fade back	1. at17=(grayscale of R 0~255, grayscale of G 0~255, grayscale of B 0~255, speed of stacking 1~30) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** at17=(0,255,100,10)	printf("at17=(%d,%d,%d,%d)",R,G,B,speed); while (USART_ReceiveData(UART1)!= 'E') {}
0xf2	Set the Dimming level *** Only available for Dimmable LEDs**	1. atf2=(Dimming level 0~31) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<example>** atf2=(7)	printf("atf2=(%d)",Dimming); while (USART_ReceiveData(UART1)!= 'E') {}

0xfd	Set the dynamic function	1. atfd=(Function Code 0~20) 2. Wait until receive a device available byte ('E') from Ring or Stripe **<Function code>** **0 : Stop the auto run back to static mode** **1~20 are auto run mode. The display speed and color can be determinated by atcc and atcd command accordingly** **1: Breath (for red, green, blue, yellow, cyan, magenta & white only)** **2: Randomly color display for whole ring** **3: Turn one pixel clockwise** **4: Turn one pixel counter clockwise** **5: Turn comet section pixels clockwise (for red, green, blue, yellow, cyan, magenta & white only)** **6: Turn comet section pixels counter clockwise (for red, green, blue, yellow, cyan, magenta & white only)** **7: Comet section pixels bounce around (for red, green, blue, yellow, cyan, magenta & white only)** **8: Turn ring display menory data clockwise (Ring display memory can be determinated by atc0 or atc1command)** **9: Turn ring display menory data counter clockwise (Ring display memory can be determinated by atc0 or atc1command)** **10: Flash whole ring** **11: Fill pixel one by one start from last pixel** **12: Fill pixel one by one start from first pixel** **13: Fill pixel one by one clockwise then disappear one by one counter clockwise** **14: Fill pixel one by one clockwise then disappear one by one clockwise** **15: Show the ring display memory data one by one then disappear one by one counter clockwise** **16: Show the ring display memory data one by one then disappear one by one clockwise** **17: Two pixels collide then firework effect** **18: Two section pixels collide then firework effect** **19: Two pixels crossing** **20: Two pixels collide then bounce back**	printf("atfd=(%d)",Function); while (USART_ReceiveData(UART1)!='E') {}