Department:

CRDC project code: **Doc ID:** **Version:** ***V 00***

Executive definition & objectives: **This document gives designer easy way to configure and use PPDB035_SPI**

Document Owner: **Engineering China**	***PPDB035_SPI application note***	Stage Gate OPEN ☐ SELECT ☐ DO ☒ IMPLEMENT ☐ PRODUCE ☐ SELL ☐ CLOSE ☐

Status	Draft ☒	In Review ☐	Released ☐
Confidential Level	External Use ☐	Internal Use ☒	Restricted ☐

	Function	**Name**	**Date**	**Signature**
Drafted by	HMI Brick Line Leader	Wilson Fei	2023-12-23	
Reviewed by				
Approved by				

Distribution		
Name	**Title**	**Department**

Internal

Revision History			
Version	**Date**	**Author**	**Modifications**
V00	2023-12-23	Wilson Fei	Initial Draft
V01	2024-1-8	Wilson Fei	Update pixel data format with RGB565 and change SPI clock polarity

Internal

Table of Content

Internal

REFERENCE

ACRONYMS & TERMINOLOGY

PPDB	Power Premium Display Brick
LCD	Liquid Crystal Display
GUI	Graphical User Interface
MCU	Micro Controller Unit. An programmable integrated circuit
SPI	Serial Peripheral Interface. A serial data communication bus
FPC	Flexible Printed Circuit

Internal

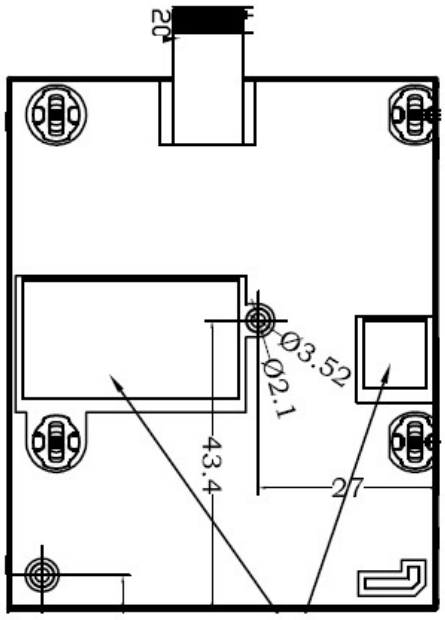
1 PPDB035_SPI OVERVIEW

PPDB035 is a 3.5inch Power Premium Display Brick, The Goal of this document is to give to the brick user all the information required to integrate the brick in its design.

PPDB035 is a LCD Module, which has 320 x 240 resolutions color TFT LCD, including a driver/controller, plastic housing and white color backlight. The electrical interface is SPI.

Table1 is the main specification of PP0B035_SPI.

Fig 1 Top view of PPDB035_SPI



Internal

Fig 2 Rear View of PPDB035_SPI

Internal

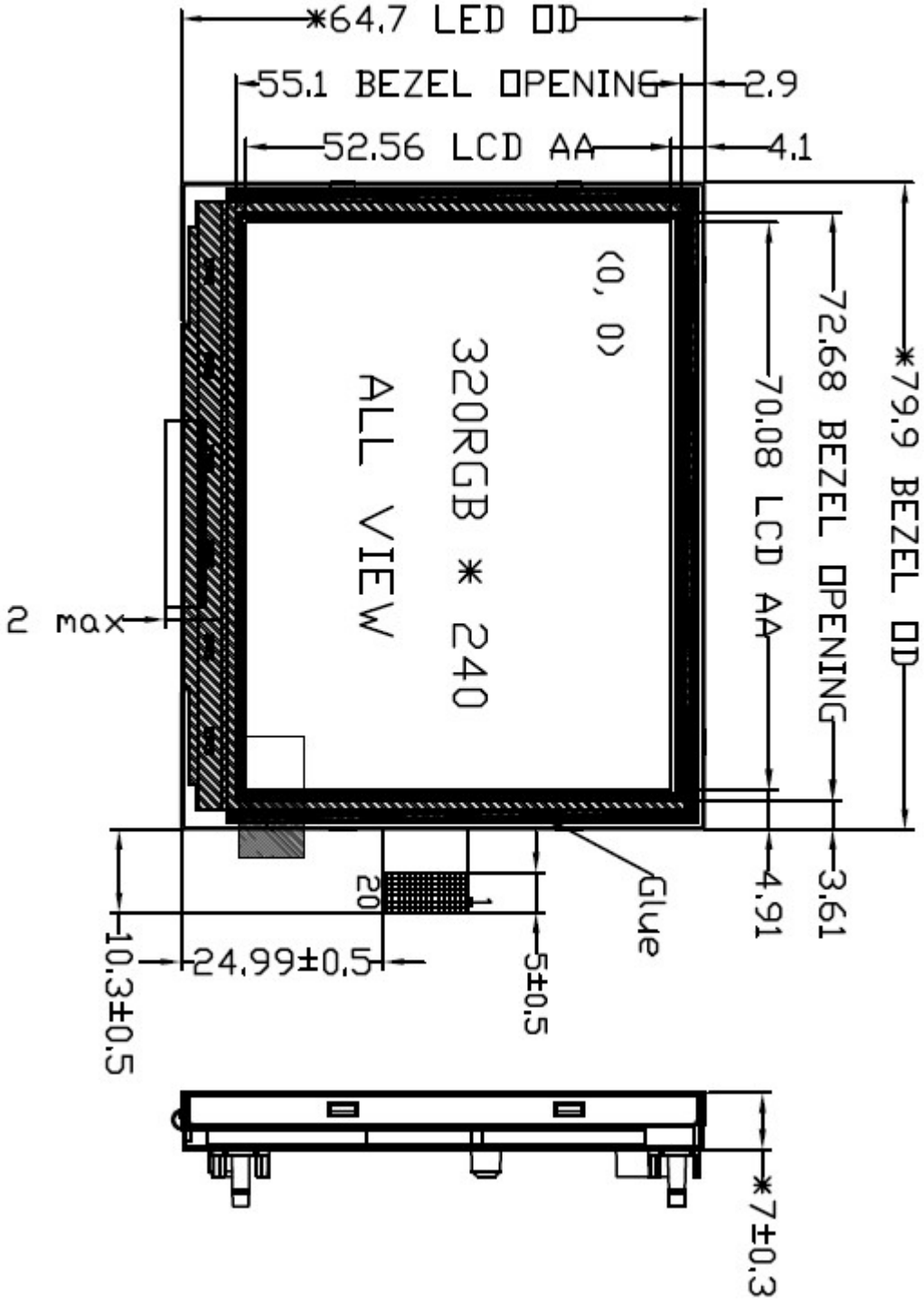


Fig3 PPDB035_SPI module drawing

Internal

Technology :	**Graphic LCD**
Resolution	320 * 240 dots
Module dimension	79.90mm x 64.70mm
Effective Viewing area (W * H mm)	72.68mm x 55.10mm
Active area (W * H mm)	70.08mm x 52.56mm
Pixel Size (W * H mm)	Square pixel, 0,23mmx0,23mm
Pixel Pitch (W * H mm)	0.219mmx0.219mm
Thickness	Less than 7,0mm(not include hook)
Mode	TFT, transmissive
Backlight Color (transmissive type)	White
Data interface	SPI
Backlight Voltage(transmissive type)	<10V
Viewing direction	6 o'clock
LCD voltage	DC/DC converter built-in
Contrast compensation	None
Operating temperature	-30°C to +80°C
Storage temperature	-40°C to +85°C
Electrical Static Discharges (ESD)	Contact Discharge ±8KV Air Discharge ±15KV IEC61000-4-2
Vibration	IEC 60068-2-6 -frequency range: 10 Hz to 150 Hz; -transition frequency: 60 Hz; -f < 60 Hz, constant amplitude of movement 0,075 mm; -f > 60 Hz, constant acceleration, 1 gn (10 m/s²) -single point control; -number of sweep cycles per axis: 10. NOTE 1 The nominal transition frequency is 60Hz, for this test the actual crossover (transition) frequency is 58,1Hz. NOTE 2 10 sweep cycles = 75 min
Humidity	Maximum 95%

Table 1: PPDB035_IPS main product specification

1.1 PPDB035_SPI Architecture

PPDB035_SPI consists of 4 parts: LCD, Driver/Controller, Backlight and FPC.

LCD is a glass panel. On this panel, LCD receives signal generating from LCD driver/controller to display certain patterns by pixels. LCD has 320RGB x 240 pixels for displaying.

LCD Driver is a general semiconductor chip but with different footprint (bare chip). The LCD driver will generate internally all the signal to drive pixels on LCD. The IC driver type is ST7272A from Sitronix.

Backlight is composed of 2*3 LEDs and optical components for emitting light from bottom of LCD, it is helpful for user to have a better view at in-door or out-door condition.

FPC is used for connection between host MCU and LCD driver/controller. Besides, it also assembles peripheral component of LCD driver.

Internal

MCU is used to store graphic data sent from user board and generate serial RGB signals to the LCD driver. The part number of the MCU is STM32U535CBT6 from ST microelectronics.

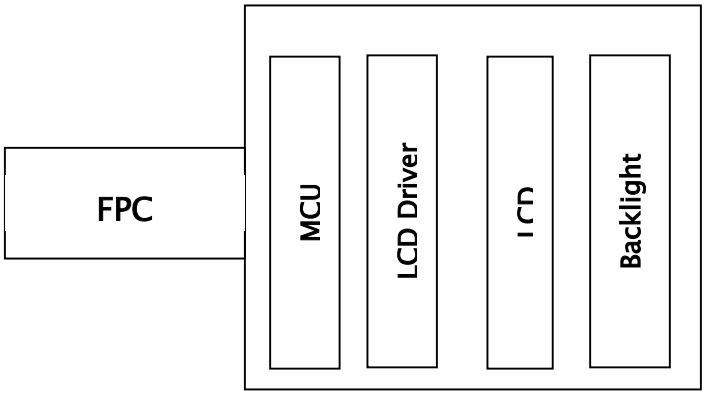


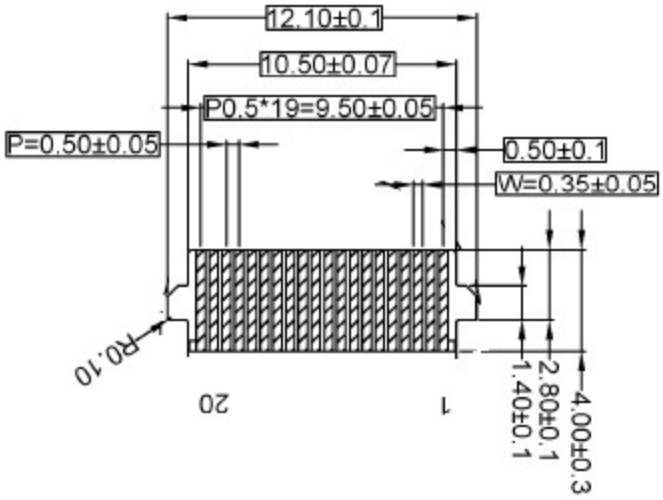
Fig4 PPDB035_SPI architecture

1.2 PPDB035_SPI Characteristics

PPDB035_SPI works at electronic conditions of VSS=0V, VDD=3.3V ±10% with operation temperature of Ta=-30 to +80℃ unless otherwise specified.

The electrical interface is a 20pin FPC terminal. Fig5 is the mechanical dimension of FPC terminal..

Table2 gives the FPC pin out definition. It is composed of backlight connection interface and SPI communication interface. MOSI, MISO and SCLK are SPI data input, SPI data output and clock pins. CS is driver IC chip select pin. A0 is for driver command and data selection pin. RESET is LCD driver reset pin. For details of SPI timing diagram please see Fig6.



Internal

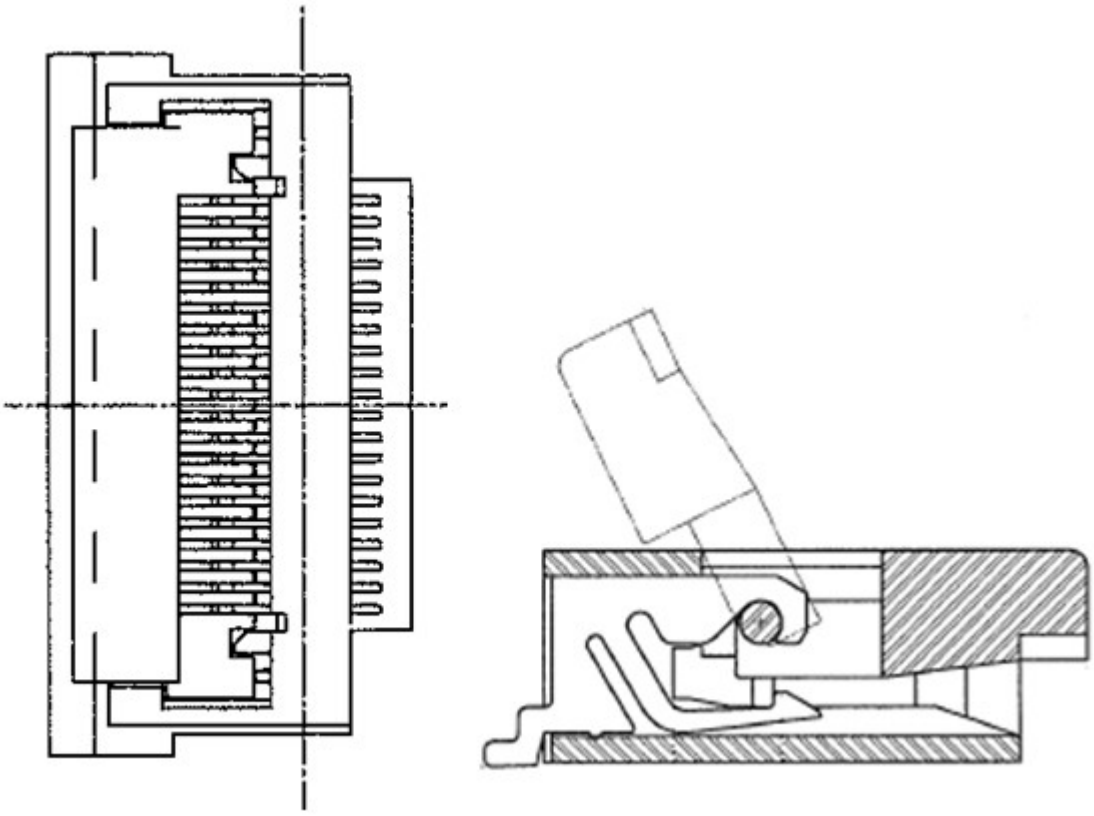


Fig5 FPC terminal and FPC connector mechanical drawing

Pin Number	Signal Name	Function
1	SPI_MISO	SPI data output
2	NC	Cathode of backlight (NC for Reflective type)
3	A1	Anode of backlight string1
4	K1	Cathode of backlight string1
5	A2	Anode of backlight string2
6	K2	Cathode of backlight string2
7	NC	No connection
8	GND	Ground (0V).
9	NC	No connection
10	VDD	Power supply (3.3V)
11	NC	No connection
12	SPI_MOSI	SPI Data input
13	GND	Ground (0V).
14	SPI_SCK	SPI clock
15	GND	Ground (0V).
16	A0	SPI Command/Data selection
17	GND	Ground (0V).
18	RESET	LCD reset (active low)
19	GND	Ground (0V).
20	CS	LCD SPI chip select (active low)

Table 2 PPDB035_SPI FPC terminal Pin definition

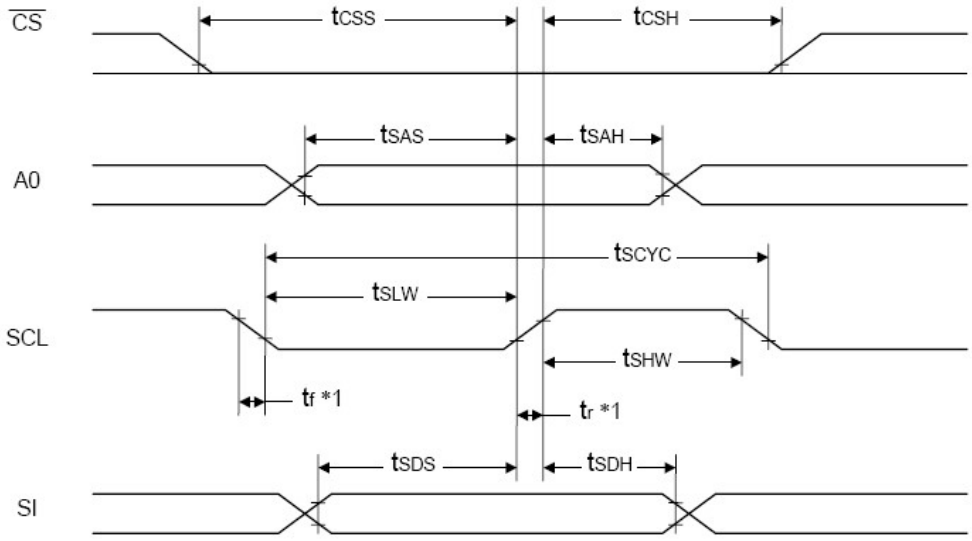
Internal

1.2.1 Electronic Characteristic.

TBD

Table 3 PPDB035_SPI Electrical Characteristic

Timing Specification:



TBD

Fig 6 SPI Timing diagram

Internal

1.2.2 Optical Characteristic

Table 4 Optical parameter

View angle

The LCD Module is normally used horizontally by 6 o'clock view. But in some applications, it is possible to be used in 12 o'clock view. Contrast and viewing angles must be adapted to such a constraint. Vertical vision angle is designed as large as possible to allow comfortable reading whatever might be the orientation of the display in the cubicle (Top or ground level).

TBD

Fig7 Viewing angle

Contrast Ratio

TBD

2 PPDB035_SPI DESIGN GUIDE

2.1 HW Design Guide

2.1.1 Backlight Circuit design

For Reflective type PPDB035_SPI, there is no backlight.

For Transmissive PPDB035_SPI, The backlight circuit is in serial with internal adjustment resistor R and parallel with Diode.

Below is the backlight LED material information

Part Number	NSSW306GT
Manufacture	NICHIA
Luminance Intensity Rank	V645~V720

Internal

Color Rank	Sa62, Sbj2
Working voltage	8.5V DC
Working current	5mA

PPDB035_SPI backlight Current-Luminance curve

Fig8a. PPDB035_SPI current-luminance curve for supplier Tianma

Fig8b. PPDB035_SPI current-luminance curve for supplier BOE-Varitronix

Backlight application circuit

The PPDB035_SPI backlight application circuit is recommended in Fig9 and Fig10. Fig9 is for consistent current design. The backlight is driven by a transistor T1. R31(260 ohm resistor) control the driving current of LED. The LED current is obtained by following equator:

I = (3.3-0.7)/R31=(3.3-0.7)/260 = 10mA.

In Fig9, a 12V DC supply is needed to drive the backlight. There will be a dedicate power system to generate 12V DC.

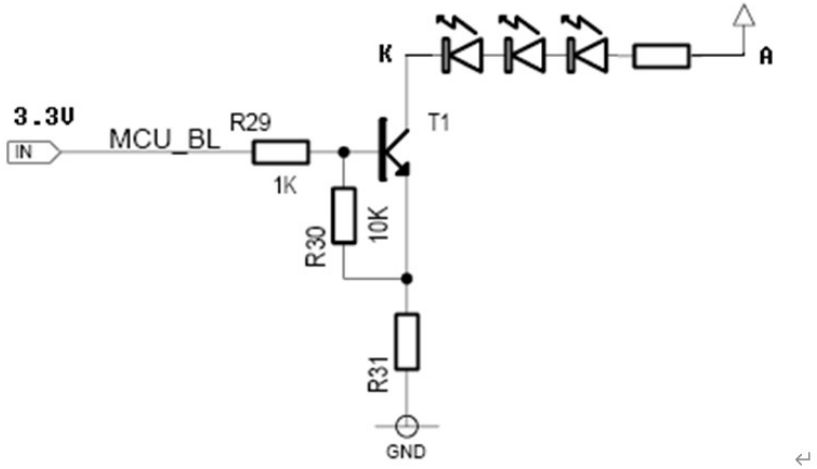


Fig 9 Consistent current design for backlight application circuit

Internal

Another method of driving the backlight is using set-up converter, like MAX1553. We can use the same voltage as the IC controller (3.3V) to drive the backlight. See Fig 10. It is also a consistent current driver. The LED current is calculated as follows. Please note that we can connect BRT as 3.3V DC or 3.3V with PWM, but in the calculation, the V_{BRT} is still taken as 1.72.

$$I_{LED} = \frac{V_{BRT} + 0.17}{6.67 \times R1} = \frac{1.72 + 0.17}{6.67 \times 28} = 10\ mA$$

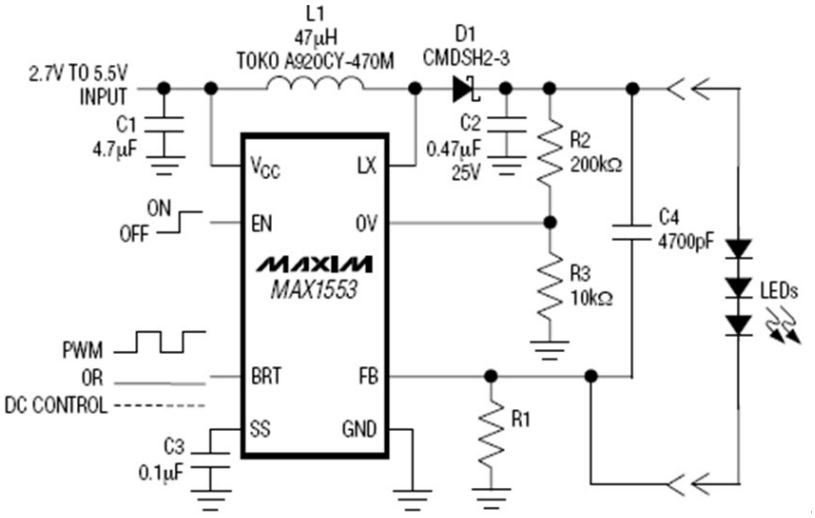


Fig10 Backlight driving with set-up converter by 3.3V input voltage

2.1.2 MCU Interface

A serial SPI interface is used for MCU reading and writing to PPDB035_SPI. An 20Pin 0.5mm pitch FPC connector is used to adapt the display FPC terminal. For detailed description of FPC connector pin definition please see Talbe5 in Section2.2.1

2.1.3 PCB layout constraint

From FPC point of view, the signal pin has been separated (shield) by GND pins. It is better to use 4-layer PCB for line routing. Keep SCLK rout enough distance (0.5mm) from other routs, or shield SCLK rout with GND.

2.2 PPDB035_SPI low level driver firmware design guide

2.2.1 SPI interface configuration

The display has a SPI interface: RESET, CS, A0, SCK, MOSI, MISO.

Internal

1) RESET(Pin18) is the LCD driver reset pin. User can pull this pin to low level to reset the LCD. When RESET pin is in low level, there is no serial RGB signal generation from the display MCU. After reset, user should keep RESET pin to high level for LCD normal operation.

2) CS (Pin20) is the chip select of SPI interface. Before sending data, CS should be pulled to low level. When there is no data to send, pull CS to high level to avoid noise inference to SPI interface.

3) A0 (Pin16) is command / data selection signal. When sending command to the display, pull A0 to low level. When sending parameters of command or graphic data, pull A0 to high level.

4) SCK (Pin14) is the clock of SPI interface. User board should be configured as SPI master. The SPI clock frequency should not exceed 20MHz due to the limitation of the MCU. To drive PPDB035_SPI, 8MHz-16MHZ SPI clock frequency is recommended. The clock polarity should be configured as active low (idle state high CPOL=1), data latching at clock second edge (rising edge, CPHA=1). Incorrect SPI configuration will cause wrong data receiving by PPDB035_SPI.

5) MOSI (Pin12) is SPI data input pin. User board transmits data through this pin.

6) MISO(Pin1) is SPI data output pin. It transmits data to the user board.

2.2.2 PPDB035_SPI programming instructions

In the electronic application system, PPDB035_SPI is connected to the host MCU through FPC connecter. The data interface is SPI. Please refer to Section 2.1.1 for SPI module configuration. The MCU can send either command or data to PPDB035_SPI LCD driver. When MCU sends command to PPDB035_SPI, the FPC connecter A0 pin is set to low voltage level. When MCU sends data (graphic data) to PPDB035_SPI, the A0 pin is set to high voltage level. In the firmware programming, we can define below functions for host MCU communicating with LCD controller.

LCD_WRITE_A0(cmd) MCU send command to LCD controller . Use this to configure LCD controller

LCD_WRITE_A1(data) MCU send command parameter or graphic data to LCD controller.

The LCD driver can be programmed by below instructions in Table5. It is highly recommended to use recommended initial code (Section 3.2.1) to drive the display. Incorrect initial code configuration will cause undetectable damage to LCD module.

Register/Command	A0 pin	Command/Data	Comments
1. Set display column	0	0x2A	
	1	0x00 - 0xEF for Portrait orientation 0x00 – 0xFF for Landscape orientation	Start column address low byte
	1	0x00 for Portrait orientation 0x00 - 0x01 for Landscape orientation	Start column address high byte
	1	0x00 - 0xEF for Portrait orientation 0x00 – 0xFF for Landscape orientation	End column address low byte
	1	0x00 for Portrait orientation 0x00 - 0x01 for Landscape orientation	End column address high byte
2. Set display row	0	0x2B	
	1	0x00 - 0xEF for Landscape orientation	Start row address low byte

Internal

		0x00 – 0xFF for Portrait orientation	
	1	0x00 for Landscape orientation 0x00 - 0x01 for Portrait orientation	Start row address high byte
	1	0x00-0xEF for Landscape orientation 0x00 - 0x01 for Portrait orientation	End row address low byte
	1	0x00 for Landscape orientation 0x00 – 0x01 for Portrait orientation	End row address high byte
3. Write data to display	0	0x2C	
	1	0x00 – 0xFF	Multiple graphic data, Data for is RGB565. 2 bytes per pixel.
5.Set display orientation	0	0x36	Set display orientation
	1	0xA0 Landscape, LCD driver on top 0x20 Landscape, IC driver on bottom 0xC0 Portrait, LCD driver on Right 0x00 Portrait, LCD driver on Left	

Table5 LCD driver instruction

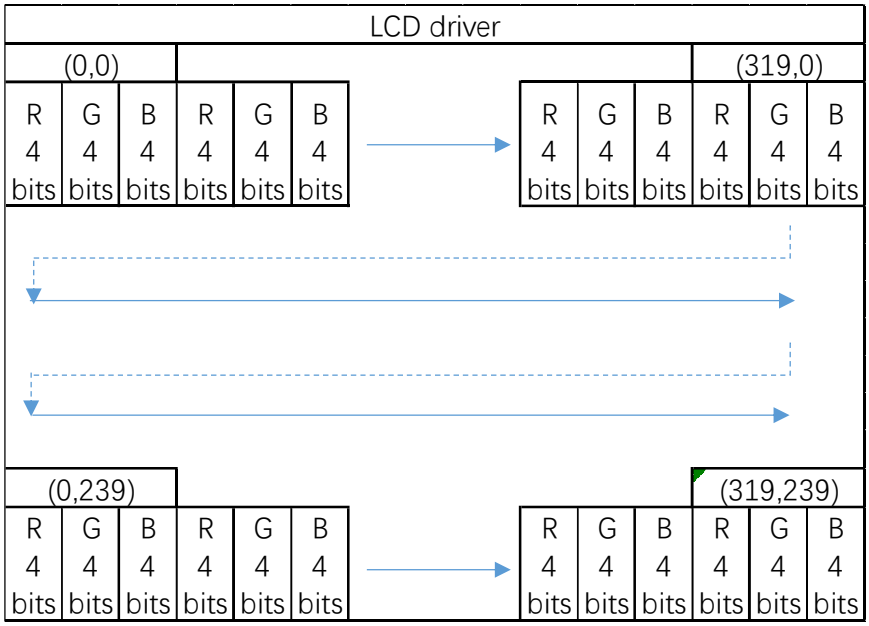
2.2.3 How to display pattern

PPDB035_SPI display only supports R-G-B 5-6-5 data format. Each pixel contains 2 bytes data with MSB bit order. Although the data format is R-G-B 5-6-5, the actual valid display color mode is R-G-B 4-4-4. See below figure.

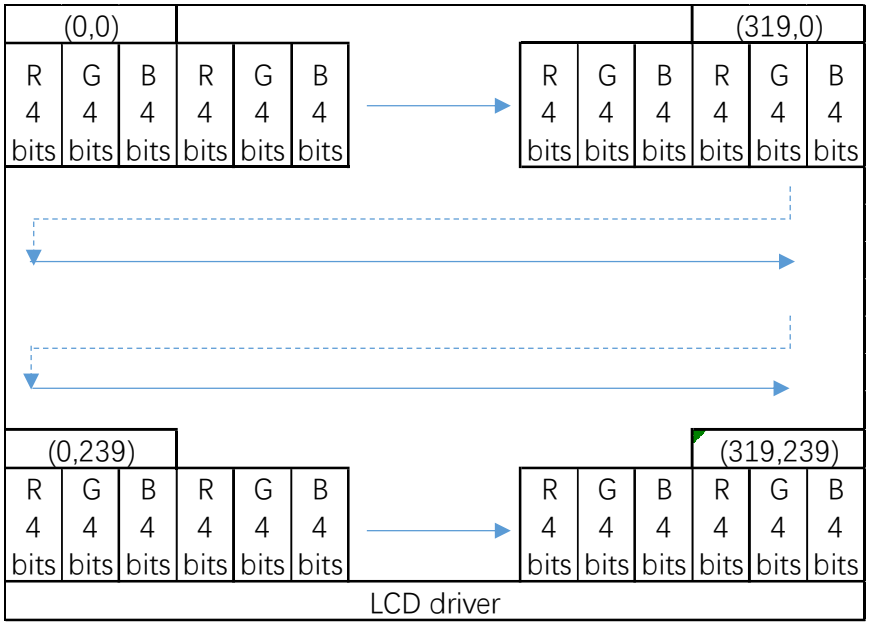
	Byte0								Byte1							
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Data Format	G2	G1	G0	B4	B3	B2	B1	B0	R4	R3	R2	R1	R0	G5	G4	G3
Valid color bit	G2			B4	B3	B2	B1		R4	R3	R2	R1		G5	G4	G3

PPDB035_SPI display is a 320RGB*240 dot matrix display. There are 4 orientation modes in the application: Landscape with LCD driver at top, Landscape with LCD driver at bottom, Portrait with LCD driver at right, Portrait with LCD driver at left. For each orientation mode, the data entry order is always from left to right and from top to bottom. See below figures.

Internal

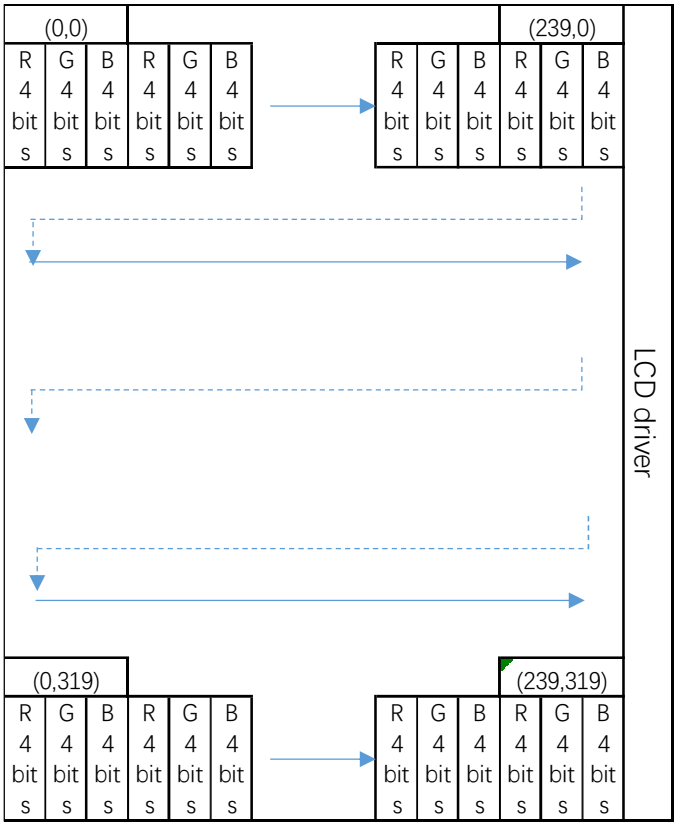


Data entry order in landscape with LCD driver on top orientation

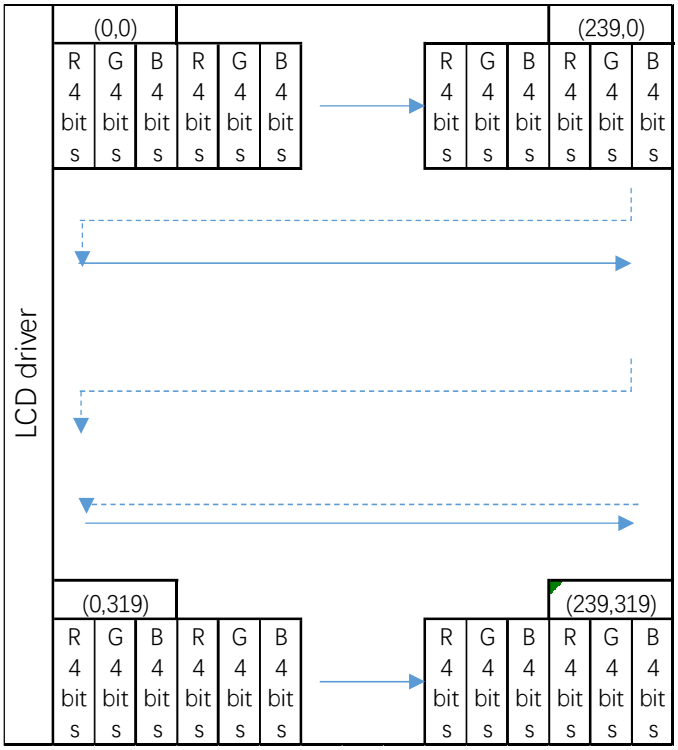


Data entry order in landscape with LCD driver on bottom orientation

Internal



Data entry order in portrait with LCD driver on right orientation



Data entry order in portrait with LCD driver on left orientation

Internal

Internal

3 PPDB035_SPI SETUP

3.1 Condition of Use

It is highly recommended for PPDB035_SPI user, discussing with your marketing seriously, whether the backlight need always on. As we know, PPDB035_SPI backlight is more like a candle, the more time you light on, the less time backlight live.

It is possible that PPDB035_SPI has multi suppliers. User can find supplier information on the display barcode label. See Fig21.

Fig21 product 2D barcode on the label

3.2 PPDB035_SPI Drive Procedure

When PPDB035_SPI is power on, a set of initial code should be executed to meet user application requirement. The purpose of these initial codes are used to configure the LCD driver internal registers. Due to different LCD suppliers LCD cell design and material choice difference, it is possible that some driver register configuration value maybe different. And that will cause some difference of initial code for different suppliers. Here gives the initial code of PPDB035_SPI.

3.2.1 Initial code

```
void LCD_Init(void)
{
  TFT_RESET_LOW; // RESET low, display off
  HAL_Delay(20); // delay 20ms
  TFT_RESET_HIGH; // RESET high, display on
  HAL_Delay(100); // delay 100ms after RESET is high

  SpiTFT_Command(0x36); // set display orientation
  SpiTFT_Data(0xB4);// Portrait mode, LCD driver on right
}
```

3.2.2 Example code

Internal

```
#define LANDSCAPE_IC_UP    0
#define LANDSCAPE_IC_DOWN  1
#define PORTRAIT_IC_UP     2
#define PORTRAIT_IC_DOWN   3

#define COLOR_MODE_RGB444  0
#define COLOR_MODE_RGB565  1


#define LCD_LAST_COLUMN         (uint16_t)319 /* 320-1 */
#define LCD_LAST_ROW            (uint16_t)239 /* 240-1 */


uint16_t gSpiBuffer[320];
uint8_t gDisplayOrientation;
uint16_t gHostWidth;
uint16_t gHostHeight;
uint16_t gX1;
uint16_t gX2;
uint16_t gY1;
uint16_t gY2;
uint16_t gY;
uint16_t gColor;
uint8_t gVHostStep;
uint32_t gDelayCounter;

uint8_t VHost_Task_RGB565(void);
void VHost_RGB565(void);
void VHost_SendCommand(uint8_t tCommand);
```


Internal

```
void VHost_SendData(uint8_t tData);

void VHost_SendDataN(uint8_t *pData,uint16_t tNum);



void VHost_SetDisplaySize(uint16_t tWidth,uint16_t tHeight);

void Init_VHost(void);
void VHost_Task(void);
uint8_t VHost_FillRect(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2,uint16_t tColor);



void VHost_SendCommand(uint8_t tCommand)
{
  LCD_Command(tCommand);
}
void VHost_SendData(uint8_t tData)
{
  LCD_Data(tData);
}
void VHost_SendDataN(uint8_t *pData,uint16_t tNum)
{
  LCD_DataN(pData,tNum);
}

void VHost_SetDisplaySize(uint16_t tWidth,uint16_t tHeight)
{
  gHostWidth=tWidth;
```


Internal

```
  gHostHeight=tHeight;
}

void Init_VHost(void)
{
  gY=1;
  gY1=0;
  gY2=0;
  gVHostStep=0;
  gDisplayOrientation=LANDSCAPE_IC_UP;
  VHost_SendCommand(0x36);
  VHost_SendData(0xA0);
  VHost_SetDisplaySize(320,240);
  //VHost_SetDisplaySize(240,320);
  //pFun_Host_Task=VHost_Task_RGB565;
}

void VHost_Task(void)
{
  VHost_RGB565();
 // VHost_RGB444();
}
void VHost_RGB565(void)
{

  switch(gVHostStep)
  {
  case 0:
    if(gDisplayOrientation==LANDSCAPE_IC_DOWN || gDisplayOrientation==LANDSCAPE_IC_UP)
```


Internal

```
  {
    VHost_SetDisplaySize(320,240);
    gVHostStep +=VHost_FillRect(0,0,319,239,0xFFFF);
  }
  else
  {
    VHost_SetDisplaySize(240,320);
    gVHostStep +=VHost_FillRect(0,0,239,319,0xFFFF);
  }
  break;
case 1:
  if(gDisplayOrientation==LANDSCAPE_IC_DOWN || gDisplayOrientation==LANDSCAPE_IC_UP)
  {
    VHost_SetDisplaySize(320,240);
    gVHostStep +=VHost_FillRect(1,1,318,238,0);
  }
  else
  {
    VHost_SetDisplaySize(240,320);
    gVHostStep +=VHost_FillRect(1,1,238,318,0);
  }
  break;
case 2:
  gVHostStep +=VHost_FillRect(10,10,50,50,0xF800);
  break;
case 3:
  gVHostStep +=VHost_FillRect(11,51,80,90,0x07E0);
  break;
case 4:
```


Internal

```
  gVHostStep +=VHost_FillRect(12,91,120,130,0x001F);
  break;
 case 5:
  gVHostStep +=VHost_FillRect(13,131,160,170,0xFFFF);
  gDelayCounter=0;
  break;
 //case 6:
  //break;
 case 6:
  gDelayCounter++;
  if(gDelayCounter>0xFF000)
  {
    VHost_SendCommand(0x36);
    //VHost_SendData(0x00);

    switch(gDisplayOrientation)
    {
    case LANDSCAPE_IC_UP:
      //VHost_SendData(0xA0);
      //gDisplayOrientation=LANDSCAPE_IC_UP;
      VHost_SendData(0x20);
      gDisplayOrientation=LANDSCAPE_IC_DOWN;
      break;
    case LANDSCAPE_IC_DOWN:
      VHost_SendData(0xC0);
      gDisplayOrientation=PORTRAIT_IC_UP;
      break;
    case PORTRAIT_IC_UP:
      VHost_SendData(0x00);
```


Internal

```
        gDisplayOrientation=PORTRAIT_IC_DOWN;
        break;
      case PORTRAIT_IC_DOWN:
        VHost_SendData(0xA0);
        gDisplayOrientation=LANDSCAPE_IC_UP;
        break;
      }

      gDelayCounter=0;
      gVHostStep=0;
    }
    break;
  }

}
uint8_t VHost_FillRect(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2,uint16_t tColor)
{
  uint8_t tResult;

  if(gY>gY2)
  {
    gX1=x1;
    gX2=x2;
    gY1=y1;
    gY2=y2;
    if(x2<x1)
    {
      gX1=x2;
      gX2=x1;
```


Internal

```
}

if(y2<y1)
{
  gY1=y2;
  gY2=y1;
}

if(gY2>gHostHeight-1)
  gY2=gHostHeight-1;
if(gX2>gHostWidth-1)
  gX2=gHostWidth-1;
if(gX1>gHostWidth-1)
  gX1=gHostWidth-1;
if(gY1>gHostHeight-1)
  gY1=gHostHeight-1;

VHost_SendCommand(0x2A);
VHost_SendData(gX1>>8);
VHost_SendData(gX1 & 0xFF);
VHost_SendData(gX2>>8);
VHost_SendData(gX2 & 0xFF);

VHost_SendCommand(0x2B);
VHost_SendData(gY1>>8);
VHost_SendData(gY1 & 0xFF);
VHost_SendData(gY2>>8);
VHost_SendData(gY2 & 0xFF);
```


Internal

```
    VHost_SendCommand(0x2C);
    gY=gY1;
    gColor=tColor;
    tResult=0;
  }
  else
    tResult= VHost_Task_RGB565();
    //tResult=(*pFun_Host_Task)();
  return tResult;
}



uint8_t VHost_Task_RGB565(void)
{
  uint16_t x;
  uint8_t tCompleted;

    if(gY<=gY2)
    {
      for(x=0;x<=gX2-gX1;x++)
      {
        gSpiBuffer[x]=gColor;
      }
      VHost_SendDataN((uint8_t *)gSpiBuffer,(gX2-gX1+1)*2);

      gY++;
    }
    if(gY>gY2)
    {
```


Internal

```
    gY=0xFFF;
    tCompleted=1;
  }
  else
  {
    tCompleted=0;
  }
  return tCompleted;

}
```

3.3 Solving Problems

After initialization, no display in PPDB035_SPI

Check your HW connections, like VDD voltage level, and SPI communication pins connections.

Check SPI communication firmware configuration parameters. Monitor SPI SCLK and SDAT signals by an oscillator. Make sure SPI communication commands are sent out correctly.

Check PPDB035_SPI initial program is executed.

No backlight

Make sure FPC connections are in good conditions.

Measure the voltage level of Pin and Pin K. The voltage should be larger than +8V.

Internal