

RAiO

RA8875

**Character/Graphic
TFT LCD Controller**

Application Note

Version 1.6

Apr, 14, 2015

Update History		
Version	Date	Description
1.0	May, 03, 2011	First Release
1.1	May,09,2011	Schematic 2 update ,initial code update
1.2	June,22,2011	Schematic 2 update
1.3	July,11,2011	Add chapter 6.appendix
1.4	Nov,30,2011	All schematic update, add serial IF,serial flash program circuit.
1.5	June,28,2013	Schematic 3 update, initial code update
1.6	Apr,14,2015	Schematic 2 update, remove MX25L256

Chapter	Contents	Page
1.	Application Circuit.....	4
2.	Initial Code.....	7
3.	Display on Sequence.....	15
4.	Sleep Mode Sequence.....	16
5.	Display RAM pure data write example	18
6.	Appendix	19

1. Application Circuit

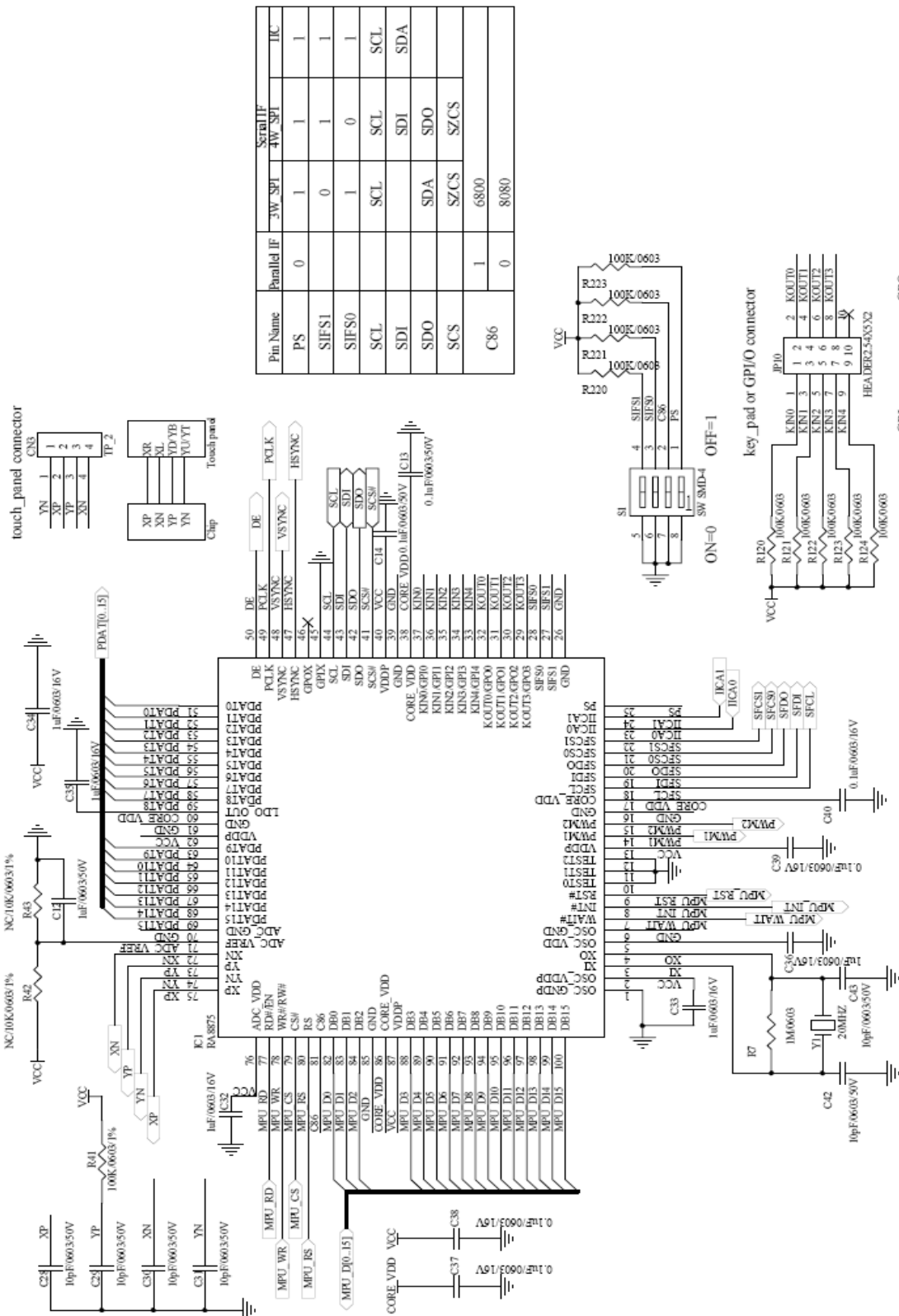
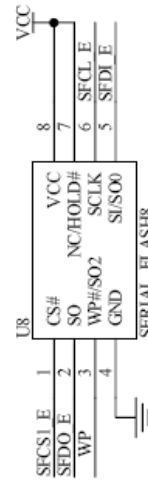
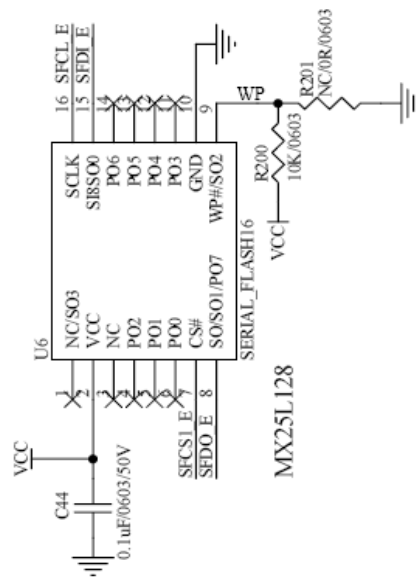


Figure 1-1

GRAPHIC SERIAL FLASH

DMA mode from serial flash to display



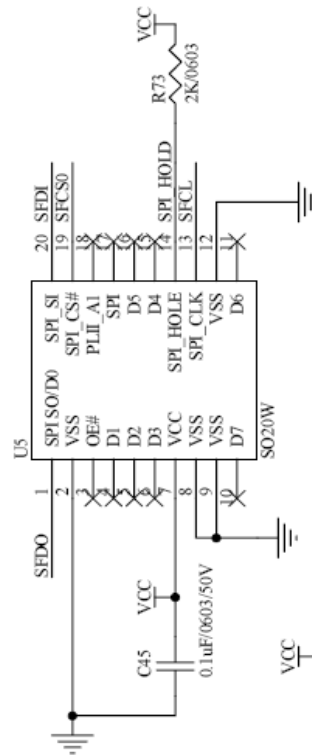
MX25L16

MX25L32

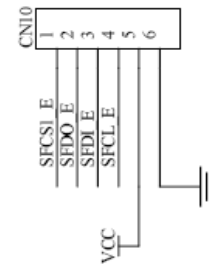
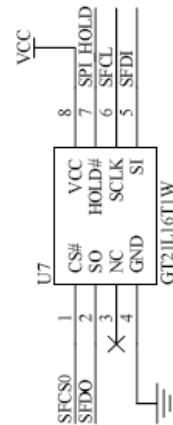
MX25L64

GENITOP FONT ROM

20PIN FOR GT23L16U2W/GTL24T3Y/GT23L24M1Z/GT23L32S4W



8PIN FOR GT21L16T1W

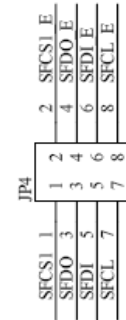
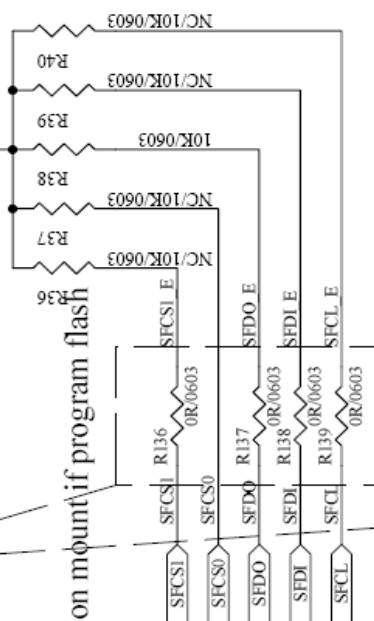


For external programmer serial flash

※(Please refer to the appendix a)

Option

Non mount if program flash



Normally operation : Short

Program flash : Open

Figure 1-2

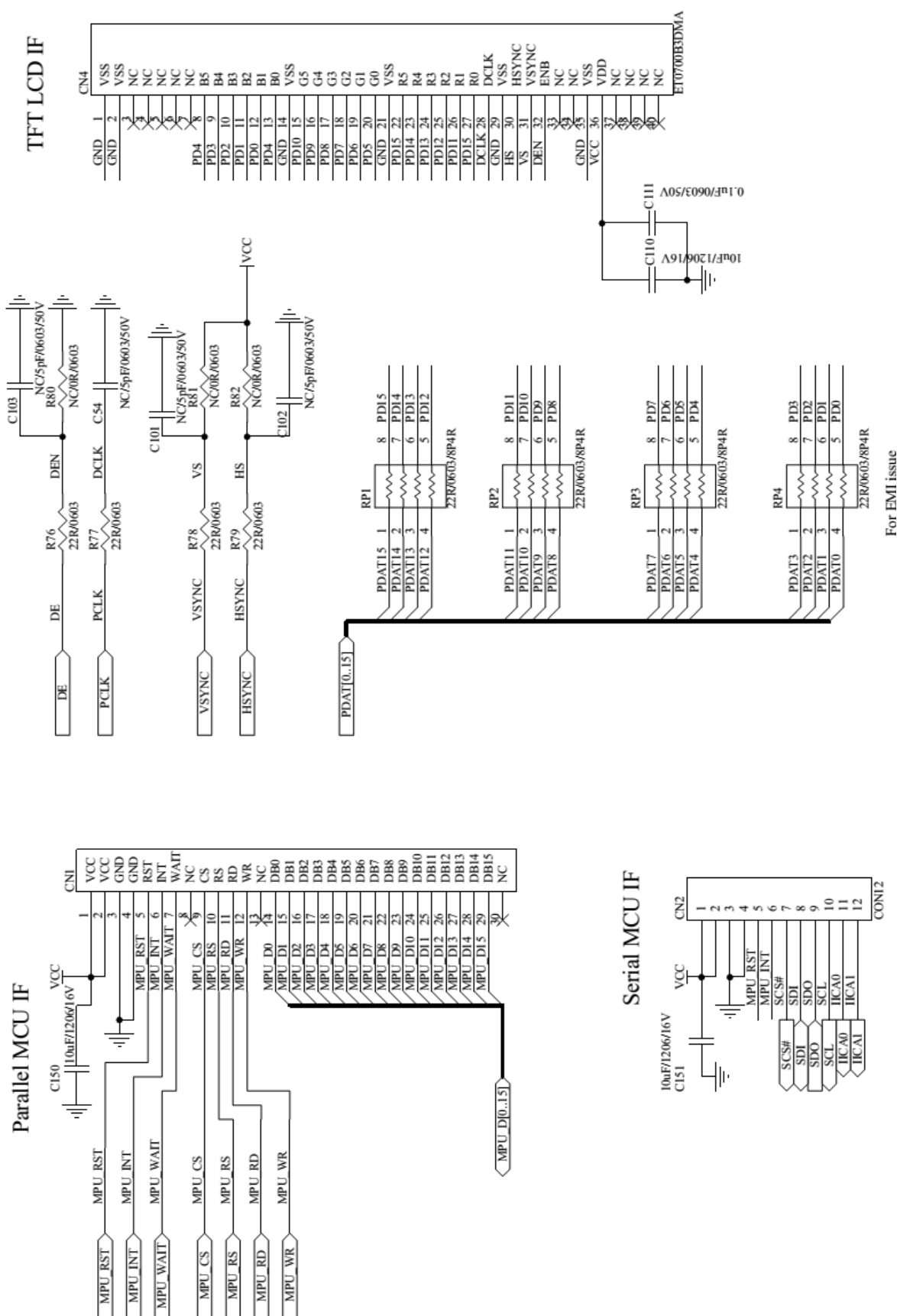


Figure 1-3

2. Initial Code

```
void RA8875_PLL_ini(void)
{ //Base on 20MHz crystal
#ifdef P320x240 //system clock =  $20 \times (10+1) / (2^2) = 55\text{MHz}$ 
    LCD_CmdWrite(0x88);
    LCD_DataWrite(0x0a);
    Delay1ms(1);
    LCD_CmdWrite(0x89);
    LCD_DataWrite(0x02);
    Delay1ms(1);
#endif

#ifdef P480x272 //system clock =  $20 \times (10+1) / (2^2) = 55\text{MHz}$ 
    LCD_CmdWrite(0x88);
    LCD_DataWrite(0x0a);
    Delay1ms(1);
    LCD_CmdWrite(0x89);
    LCD_DataWrite(0x02);
    Delay1ms(1);
#endif

#ifdef P640x480 //system clock =  $20 \times (11+1) / (2^2) = 60\text{MHz}$ 
    LCD_CmdWrite(0x88);
    LCD_DataWrite(0x0b);
    Delay1ms(1);
    LCD_CmdWrite(0x89);
    LCD_DataWrite(0x02);
    Delay1ms(1);
#endif

#ifdef P800x480 //system clock =  $20 \times (11+1) / (2^2) = 60\text{MHz}$ 
    LCD_CmdWrite(0x88);
    LCD_DataWrite(0x0b);
    Delay1ms(1);
    LCD_CmdWrite(0x89);
    LCD_DataWrite(0x02);
    Delay1ms(1);
#endif
}
```

#endif

}

//-----//

void LCD_Initial(void)

{

RA8875_PLL_ini();

LCD_CmdWrite(0x10); //SYSR bit[4:3]=00 256 color bit[2:1]= 00 8bit MPU interface

LCD_DataWrite(0x0c); // if 8bit MCU interface and 65k color display

//LCD_DataWrite(0x0F); // if 16bit MCU interface and 65k color display

#ifdef P320x240

//===== Display Window320x240 =====

LCD_CmdWrite(0x04); //set PCLK invers

LCD_DataWrite(0x03); //PCLK = system clock/(2^3) = 55/8 = 6.875MHz

Delay1ms(1);

//Horizontal set

LCD_CmdWrite(0x14); //HDWR//Horizontal Display Width Setting Bit[6:0]

LCD_DataWrite(0x27); //Horizontal display width(pixels) = (HDWR + 1)*8

LCD_CmdWrite(0x15); //Horizontal Non-Display Period Fine Tuning Option Register (HNDFT)

LCD_DataWrite(0x00); //Horizontal Non-Display Period Fine Tuning(HNDFT) [3:0]

LCD_CmdWrite(0x16); //HNDR//Horizontal Non-Display Period Bit[4:0]

LCD_DataWrite(0x05); //Horizontal Non-Display Period (pixels) = (HNDR + 1)*8

LCD_CmdWrite(0x17); //HSTR//HSYNC Start Position[4:0]

LCD_DataWrite(0x04); //HSYNC Start Position(PCLK) = (HSTR + 1)*8

LCD_CmdWrite(0x18); //HPWR//HSYNC Polarity ,The period width of HSYNC.

LCD_DataWrite(0x03); //HSYNC Width [4:0] HSYNC Pulse width(PCLK) = (HPWR + 1)*8

//Vertical set

LCD_CmdWrite(0x19); //VDHR0 //Vertical Display Height Bit [7:0]

LCD_DataWrite(0xef); //Vertical pixels = VDHR + 1

LCD_CmdWrite(0x1A); //VDHR1 //Vertical Display Height Bit [8]

LCD_DataWrite(0x00); //Vertical pixels = VDHR + 1

LCD_CmdWrite(0x1B); //VNDR0 //Vertical Non-Display Period Bit [7:0]

LCD_DataWrite(0x05); //Vertical Non-Display area = (VNDR + 1)

LCD_CmdWrite(0x1C); //VNDR1 //Vertical Non-Display Period Bit [8]

LCD_DataWrite(0x00); //Vertical Non-Display area = (VNDR + 1)

LCD_CmdWrite(0x1D); //VSTR0 //VSYNC Start Position[7:0]

LCD_DataWrite(0x0e); //VSYNC Start Position(PCLK) = (VSTR + 1)

LCD_CmdWrite(0x1E); //VSTR1 //VSYNC Start Position[8]

LCD_DataWrite(0x00); //VSYNC Start Position(PCLK) = (VSTR + 1)

```
LCD_CmdWrite(0x1F); //VPWR //VSYNC Polarity ,VSYNC Pulse Width[6:0]
LCD_DataWrite(0x02); //VSYNC Pulse Width(PCLK) = (VPWR + 1)
```

```
//Active window set
```

```
//setting active window X
```

```
LCD_CmdWrite(0x30); //Horizontal Start Point 0 of Active Window (HSAW0)
LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [7:0]
LCD_CmdWrite(0x31); //Horizontal Start Point 1 of Active Window (HSAW1)
LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [9:8]
LCD_CmdWrite(0x34); //Horizontal End Point 0 of Active Window (HEAW0)
LCD_DataWrite(0x3F); //Horizontal End Point of Active Window [7:0]
LCD_CmdWrite(0x35); //Horizontal End Point 1 of Active Window (HEAW1)
LCD_DataWrite(0x01); //Horizontal End Point of Active Window [9:8]
```

```
//setting active window Y
```

```
LCD_CmdWrite(0x32); //Vertical Start Point 0 of Active Window (VSAW0)
LCD_DataWrite(0x00); //Vertical Start Point of Active Window [7:0]
LCD_CmdWrite(0x33); //Vertical Start Point 1 of Active Window (VSAW1)
LCD_DataWrite(0x00); //Vertical Start Point of Active Window [8]
LCD_CmdWrite(0x36); //Vertical End Point of Active Window 0 (VEAW0)
LCD_DataWrite(0xef); //Vertical End Point of Active Window [7:0]
LCD_CmdWrite(0x37); //Vertical End Point of Active Window 1 (VEAW1)
LCD_DataWrite(0x00); //Vertical End Point of Active Window [8]
```

```
#endif
```

```
#ifdef P480x272
```

```
LCD_CmdWrite(0x04); //set PCLK invers
LCD_DataWrite(0x82); //PCLK = system clock/(2^2) = 55/4 = 13.75 MHz
Delay1ms(1);
```

```
//Horizontal set
```

```
LCD_CmdWrite(0x14); //HDWR//Horizontal Display Width Setting Bit[6:0]
LCD_DataWrite(0x3B); //Horizontal display width(pixels) = (HDWR + 1)*8
LCD_CmdWrite(0x15); //Horizontal Non-Display Period Fine Tuning Option Register (HNDFT)
LCD_DataWrite(0x00); //Horizontal Non-Display Period Fine Tuning(HNDFT) [3:0]
LCD_CmdWrite(0x16); //HNDR//Horizontal Non-Display Period Bit[4:0]
LCD_DataWrite(0x01); //Horizontal Non-Display Period (pixels) = (HNDR + 1)*8
LCD_CmdWrite(0x17); //HSTR//HSYNC Start Position[4:0]
LCD_DataWrite(0x00); //HSYNC Start Position(PCLK) = (HSTR + 1)*8
```

LCD_CmdWrite(0x18); //HPWR//HSYNC Polarity ,The period width of HSYNC.

LCD_DataWrite(0x05); //HSYNC Width [4:0] HSYNC Pulse width(PCLK) = (HPWR + 1)*8

//Vertical set

LCD_CmdWrite(0x19); //VDHR0 //Vertical Display Height Bit [7:0]

LCD_DataWrite(0x0f); //Vertical pixels = VDHR + 1

LCD_CmdWrite(0x1a); //VDHR1 //Vertical Display Height Bit [8]

LCD_DataWrite(0x01); //Vertical pixels = VDHR + 1

LCD_CmdWrite(0x1b); //VNDR0 //Vertical Non-Display Period Bit [7:0]

LCD_DataWrite(0x02); //VSYNC Start Position(PCLK) = (VSTR + 1)

LCD_CmdWrite(0x1c); //VNDR1 //Vertical Non-Display Period Bit [8]

LCD_DataWrite(0x00); //Vertical Non-Display area = (VNDR + 1)

LCD_CmdWrite(0x1d); //VSTR0 //VSYNC Start Position[7:0]

LCD_DataWrite(0x07); //VSYNC Start Position(PCLK) = (VSTR + 1)

LCD_CmdWrite(0x1e); //VSTR1 //VSYNC Start Position[8]

LCD_DataWrite(0x00); //VSYNC Start Position(PCLK) = (VSTR + 1)

LCD_CmdWrite(0x1f); //VPWR //VSYNC Polarity ,VSYNC Pulse Width[6:0]

LCD_DataWrite(0x09); //VSYNC Pulse Width(PCLK) = (VPWR + 1)

//Active window set

//setting active window X

LCD_CmdWrite(0x30); //Horizontal Start Point 0 of Active Window (HSAW0)

LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [7:0]

LCD_CmdWrite(0x31); //Horizontal Start Point 1 of Active Window (HSAW1)

LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [9:8]

LCD_CmdWrite(0x34); //Horizontal End Point 0 of Active Window (HEAW0)

LCD_DataWrite(0xDF); //Horizontal End Point of Active Window [7:0]

LCD_CmdWrite(0x35); //Horizontal End Point 1 of Active Window (HEAW1)

LCD_DataWrite(0x01); //Horizontal End Point of Active Window [9:8]

//setting active window Y

LCD_CmdWrite(0x32); //Vertical Start Point 0 of Active Window (VSAW0)

LCD_DataWrite(0x00); //Vertical Start Point of Active Window [7:0]

LCD_CmdWrite(0x33); //Vertical Start Point 1 of Active Window (VSAW1)

LCD_DataWrite(0x00); //Vertical Start Point of Active Window [8]

LCD_CmdWrite(0x36); //Vertical End Point of Active Window 0 (VEAW0)

LCD_DataWrite(0x0F); //Vertical End Point of Active Window [7:0]

LCD_CmdWrite(0x37); //Vertical End Point of Active Window 1 (VEAW1)

LCD_DataWrite(0x01); //Vertical End Point of Active Window [8]

#endif

#ifdef P640x480

//===== Display Window640x480 =====

```
LCD_CmdWrite(0x04);    //PCLK inverse
LCD_DataWrite(0x01);    //PCLK = system clock/2 = 60/2 = 30 MHz
Delay1ms(1);
//Horizontal set
LCD_CmdWrite(0x14);//HDWR//Horizontal Display Width Setting Bit[6:0]
LCD_DataWrite(0x4F);//Horizontal display width(pixels) = (HDWR + 1)*8
LCD_CmdWrite(0x15); //Horizontal Non-Display Period Fine Tuning Option Register (HNDFTR)
LCD_DataWrite(0x05);//Horizontal Non-Display Period Fine Tuning(HNDFT) [3:0]
LCD_CmdWrite(0x16);//HNDR//Horizontal Non-Display Period Bit[4:0]
LCD_DataWrite(0x0f);//Horizontal Non-Display Period (pixels) = (HNDR + 1)*8
LCD_CmdWrite(0x17);//HSTR//HSYNC Start Position[4:0]
LCD_DataWrite(0x01);//HSYNC Start Position(PCLK) = (HSTR + 1)*8
LCD_CmdWrite(0x18); //HPWR//HSYNC Polarity ,The period width of HSYNC.
LCD_DataWrite(0x00);//HSYNC Width [4:0]    HSYNC Pulse width(PCLK) = (HPWR + 1)*8
//Vertical set
LCD_CmdWrite(0x19); //VDHR0 //Vertical Display Height Bit [7:0]
LCD_DataWrite(0xdf);//Vertical pixels = VDHR + 1
LCD_CmdWrite(0x1A);//VDHR1 //Vertical Display Height Bit [8]
LCD_DataWrite(0x01);//Vertical pixels = VDHR + 1
LCD_CmdWrite(0x1B);//VNDR0 //Vertical Non-Display Period Bit [7:0]
LCD_DataWrite(0x0A);//Vertical Non-Display area = (VNDR + 1)
LCD_CmdWrite(0x1C);//VNDR1 //Vertical Non-Display Period Bit [8]
LCD_DataWrite(0x00);//Vertical Non-Display area = (VNDR + 1)
LCD_CmdWrite(0x1D);//VSTR0 //VSYNC Start Position[7:0]
LCD_DataWrite(0x0E);//VSYNC Start Position(PCLK) = (VSTR + 1)
LCD_CmdWrite(0x1E);//VSTR1 //VSYNC Start Position[8]
LCD_DataWrite(0x00);//VSYNC Start Position(PCLK) = (VSTR + 1)
LCD_CmdWrite(0x1F);//VPWR //VSYNC Polarity ,VSYNC Pulse Width[6:0]
LCD_DataWrite(0x01);//VSYNC Pulse Width(PCLK) = (VPWR + 1)
```

//Active window set

//setting active window X

```
LCD_CmdWrite(0x30); //Horizontal Start Point 0 of Active Window (HSAW0)
LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [7:0]
LCD_CmdWrite(0x31); //Horizontal Start Point 1 of Active Window (HSAW1)
LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [9:8]
```

```
LCD_CmdWrite(0x34); //Horizontal End Point 0 of Active Window (HEAW0)
LCD_DataWrite(0x7f); //Horizontal End Point of Active Window [7:0]
LCD_CmdWrite(0x35); //Horizontal End Point 1 of Active Window (HEAW1)
LCD_DataWrite(0x02); //Horizontal End Point of Active Window [9:8]
```

```
//setting active window Y
```

```
LCD_CmdWrite(0x32); //Vertical Start Point 0 of Active Window (VSAW0)
LCD_DataWrite(0x00); //Vertical Start Point of Active Window [7:0]
LCD_CmdWrite(0x33); //Vertical Start Point 1 of Active Window (VSAW1)
LCD_DataWrite(0x00); //Vertical Start Point of Active Window [8]
LCD_CmdWrite(0x36); //Vertical End Point of Active Window 0 (VEAW0)
LCD_DataWrite(0xdf); //Vertical End Point of Active Window [7:0]
LCD_CmdWrite(0x37); //Vertical End Point of Active Window 1 (VEAW1)
LCD_DataWrite(0x01); //Vertical End Point of Active Window [8]
```

```
#endif
```

```
#ifdef P800x480
```

```
//AT070TN92 setting
```

```
//===== Display Window800x480 =====
```

```
/*
```

```
LCD_CmdWrite(0x04);    //PCLK inverse
LCD_DataWrite(0x81);    //PCLK = system clock/2 = 60/2 = 30 MHz
Delay1ms(1);
//Horizontal set
LCD_CmdWrite(0x14); //HDWR//Horizontal Display Width Setting Bit[6:0]
LCD_DataWrite(0x63); //Horizontal display width(pixels) = (HDWR + 1)*8
LCD_CmdWrite(0x15); //Horizontal Non-Display Period Fine Tuning Option Register (HNDFT)
LCD_DataWrite(0x03); //Horizontal Non-Display Period Fine Tuning(HNDFT) [3:0]
LCD_CmdWrite(0x16); //HNDR//Horizontal Non-Display Period Bit[4:0]
LCD_DataWrite(0x03); //Horizontal Non-Display Period (pixels) = (HNDR + 1)*8
LCD_CmdWrite(0x17); //HSTR//HSYNC Start Position[4:0]
LCD_DataWrite(0x02); //HSYNC Start Position(PCLK) = (HSTR + 1)*8
LCD_CmdWrite(0x18); //HPWR//HSYNC Polarity ,The period width of HSYNC.
LCD_DataWrite(0x00); //HSYNC Width [4:0]   HSYNC Pulse width(PCLK) = (HPWR + 1)*8
//Vertical set
LCD_CmdWrite(0x19); //VDHR0 //Vertical Display Height Bit [7:0]
LCD_DataWrite(0xdf); //Vertical pixels = VDHR + 1
```

```
LCD_CmdWrite(0x1a); //VDHR1 //Vertical Display Height Bit [8]
LCD_DataWrite(0x01); //Vertical pixels = VDHR + 1
LCD_CmdWrite(0x1b); //VNDR0 //Vertical Non-Display Period Bit [7:0]
LCD_DataWrite(0x14); //Vertical Non-Display area = (VNDR + 1)
LCD_CmdWrite(0x1c); //VNDR1 //Vertical Non-Display Period Bit [8]
LCD_DataWrite(0x00); //Vertical Non-Display area = (VNDR + 1)
LCD_CmdWrite(0x1d); //VSTR0 //VSYNC Start Position[7:0]
LCD_DataWrite(0x06); //VSYNC Start Position(PCLK) = (VSTR + 1)
LCD_CmdWrite(0x1e); //VSTR1 //VSYNC Start Position[8]
LCD_DataWrite(0x00); //VSYNC Start Position(PCLK) = (VSTR + 1)
LCD_CmdWrite(0x1f); //VPWR //VSYNC Polarity ,VSYNC Pulse Width[6:0]
LCD_DataWrite(0x01); //VSYNC Pulse Width(PCLK) = (VPWR + 1)
//LCD_CmdWrite(0xf2);
//LCD_DataWrite(0x01);
*/
//HSD050IDW1 setting
//===== Display Window800x480 =====
LCD_CmdWrite(0x04); //PCLK inverse
LCD_DataWrite(0x81);
Delay1ms(1);
//Horizontal set
LCD_CmdWrite(0x14); //HDWR//Horizontal Display Width Setting Bit[6:0]
LCD_DataWrite(0x63); //Horizontal display width(pixels) = (HDWR + 1)*8
LCD_CmdWrite(0x15); //Horizontal Non-Display Period Fine Tuning Option Register (HNDFTR)
LCD_DataWrite(0x00); //Horizontal Non-Display Period Fine Tuning(HNDFT) [3:0]
LCD_CmdWrite(0x16); //HNDR//Horizontal Non-Display Period Bit[4:0]
LCD_DataWrite(0x03); //Horizontal Non-Display Period (pixels) = (HNDR + 1)*8
LCD_CmdWrite(0x17); //HSTR//HSYNC Start Position[4:0]
LCD_DataWrite(0x03); //HSYNC Start Position(PCLK) = (HSTR + 1)*8
LCD_CmdWrite(0x18); //HPWR//HSYNC Polarity ,The period width of HSYNC.
LCD_DataWrite(0x0B); //HSYNC Width [4:0] HSYNC Pulse width(PCLK) = (HPWR + 1)*8
//Vertical set
LCD_CmdWrite(0x19); //VDHR0 //Vertical Display Height Bit [7:0]
LCD_DataWrite(0xdf); //Vertical pixels = VDHR + 1
LCD_CmdWrite(0x1a); //VDHR1 //Vertical Display Height Bit [8]
LCD_DataWrite(0x01); //Vertical pixels = VDHR + 1
LCD_CmdWrite(0x1b); //VNDR0 //Vertical Non-Display Period Bit [7:0]
LCD_DataWrite(0x20); //Vertical Non-Display area = (VNDR + 1)
LCD_CmdWrite(0x1c); //VNDR1 //Vertical Non-Display Period Bit [8]
LCD_DataWrite(0x00); //Vertical Non-Display area = (VNDR + 1)
```

```
LCD_CmdWrite(0x1d); //VSTR0 //VSYNC Start Position[7:0]
LCD_DataWrite(0x16); //VSYNC Start Position(PCLK) = (VSTR + 1)
LCD_CmdWrite(0x1e); //VSTR1 //VSYNC Start Position[8]
LCD_DataWrite(0x00); //VSYNC Start Position(PCLK) = (VSTR + 1)
LCD_CmdWrite(0x1f); //VPWR //VSYNC Polarity ,VSYNC Pulse Width[6:0]
LCD_DataWrite(0x01); //VSYNC Pulse Width(PCLK) = (VPWR + 1)

//Active window  set
//setting active window X
LCD_CmdWrite(0x30); //Horizontal Start Point 0 of Active Window (HSAW0)
LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [7:0]
LCD_CmdWrite(0x31); //Horizontal Start Point 1 of Active Window (HSAW1)
LCD_DataWrite(0x00); //Horizontal Start Point of Active Window [9:8]
LCD_CmdWrite(0x34); //Horizontal End Point 0 of Active Window (HEAW0)
LCD_DataWrite(0x1F); //Horizontal End Point of Active Window [7:0]
LCD_CmdWrite(0x35); //Horizontal End Point 1 of Active Window (HEAW1)
LCD_DataWrite(0x03); //Horizontal End Point of Active Window [9:8]

//setting active window Y
LCD_CmdWrite(0x32); //Vertical Start Point 0 of Active Window (VSAW0)
LCD_DataWrite(0x00); //Vertical Start Point of Active Window [7:0]
LCD_CmdWrite(0x33); //Vertical Start Point 1 of Active Window (VSAW1)
LCD_DataWrite(0x00); //Vertical Start Point of Active Window [8]
LCD_CmdWrite(0x36); //Vertical End Point of Active Window 0 (VEAW0)
LCD_DataWrite(0xdf); //Vertical End Point of Active Window [7:0]
LCD_CmdWrite(0x37); //Vertical End Point of Active Window 1 (VEAW1)
LCD_DataWrite(0x01); //Vertical End Point of Active Window [8]

#endif
}
```

3. Display on Sequence

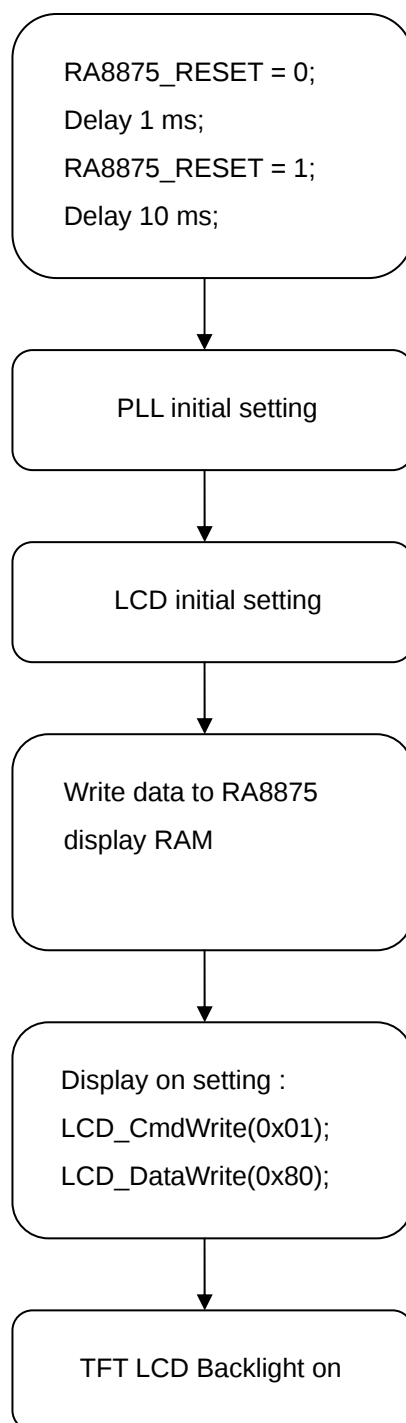


Figure 3-1

4. Sleep Mode Sequence

Enter Sleep :

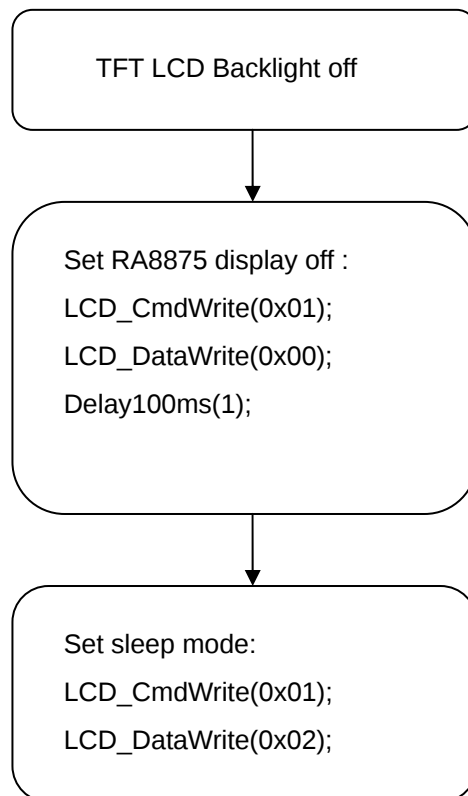


Figure 4-1

***Attention:**

We suggest to confirm your TFT panel could be disable when the RA8875 enter sleep, to avoid **liquid crystal polarization**.RAiO shall not be held liable for any damages about TFT panel ,when customers use sleep mode incorrect !

Exit Sleep :

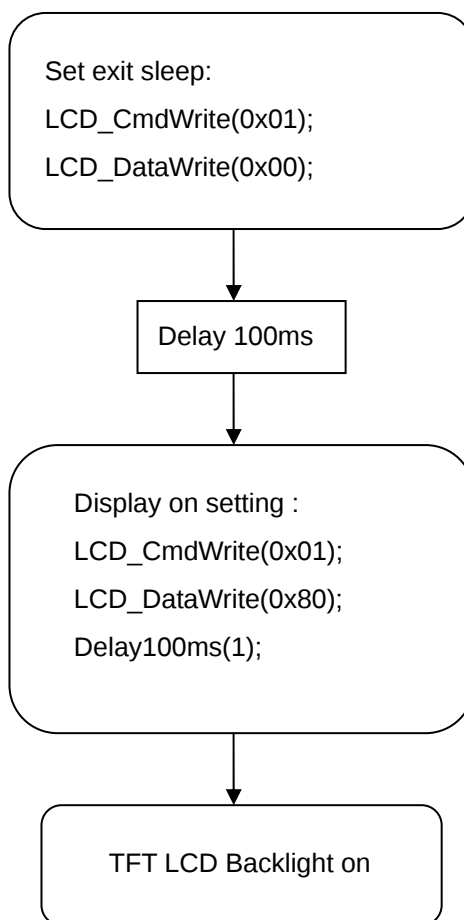


Figure 4-2

5. Display RAM pure data write example

Include RA8875_subroutine.c first , then refer the code below.

If MCU 8 bit interface :

```
Graphic_Mode( ); //set to graphic mode
XY_Coordinate(0,0); //set write cursor position
LCD_CmdWrite(0x02); //set CMD [02h] before data write
for (i=384000;i>0;i--)
{ LCD_DataWrite(0xf8); // write color red data
  LCD_DataWrite(0x00);
}
```

If MCU 16 bit interface :

```
Graphic_Mode( ); //set to graphic mode
XY_Coordinate(0,0); //set write cursor position
LCD_CmdWrite(0x02); //set CMD [02h] before data write
for (i=384000;i>0;i--)
{ LCD_DataWrite(0xf800); } // write color red data
```

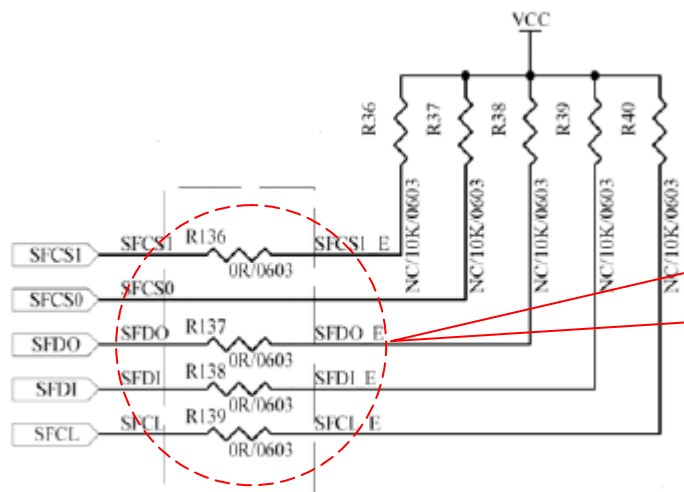
You could download some example code and RA8875 subroutine on the website :

www.raio.com.tw

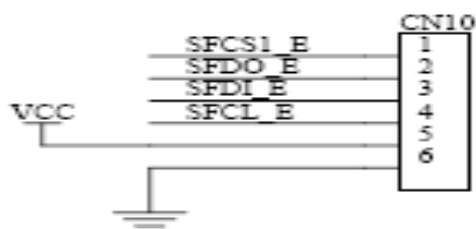
6. Appendix

a. How to program the Serial Flash Memory by the external programmer?

If the advance circuit is already designed by the LCM factory as the following picture1 and picture2, just leading the related pins of the serial flash out to the connector and then the user will be able to program the picture data(*.bin) by the external memory programmer. Please refer to the following picture3.

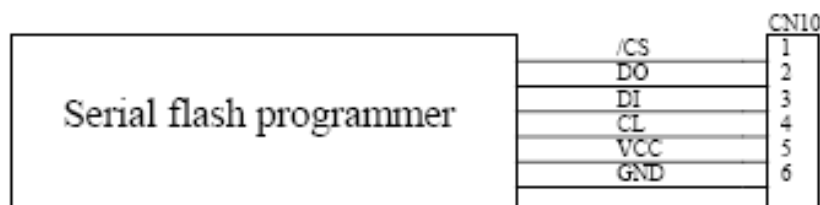


Picture 1



For external programmer serial flash

Picture2



Picture3