

**AT32 LCD Touch Driven By SPI****Introduction**

This application note introduces a general method to implement SPI-driven LCD touch screen based on AT32 MCUs.

Applicable products:

Part number	AT32F403A series
	AT32F407 series

## Contents

<b>1</b>	<b>Introduction to LCD touch screen.....</b>	<b>5</b>
1.1	Touch sensor .....	6
1.2	LCD display .....	7
<b>2</b>	<b>Operating principles .....</b>	<b>8</b>
2.1	Analog input .....	8
2.2	Internal reference voltage .....	8
2.3	External reference voltage input .....	9
2.4	Differential mode .....	9
2.5	Software operation .....	10
<b>3</b>	<b>How to use LCD touch screen .....</b>	<b>12</b>
3.1	Hardware .....	12
3.2	Software .....	12
3.3	Main codes .....	12
3.4	LCD Touch demo .....	17
<b>4</b>	<b>Revision history .....</b>	<b>20</b>

## List of tables

Table 1. Document revision history.....	20
---	----

## List of figures

Figure 1. LCD touch screen structure.....	5
Figure 2. LCD touch screen equivalent circuit.....	5
Figure 3. XPT2046 block diagram .....	7
Figure 4. Analog input loop .....	8
Figure 5. Internal voltage source .....	9
Figure 6. Reference voltage source in differential mode.....	10
Figure 7. Software operation flow .....	11
Figure 8. AT-START-F403A V1.2 evaluation board and LCD touch.....	12
Figure 9. Touch screen adjust interface.....	18
Figure 10. Touch screen adjust information.....	18
Figure 11. Touch screen test interface.....	19

# 1 Introduction to LCD touch screen

The structure of 4-wire resistive touch screen is shown in Figure 1. The glass or acrylic substrate is covered with two transparent and uniformly conductive ITO layers that are separated by evenly arranged transparent spacer dots, serving as the X electrode and Y electrode, respectively. The bottom ITO layer is attached to the glass substrate, while the top ITO layer adheres to the PET film. The anode and cathode of the X and Y electrodes are led by conductive bars (the black bars in Figure 1), and the conductive bars of X and Y electrodes are perpendicular to each other, namely the X-, X+, Y- and Y+. When an object touches the surface of touch screen and gives a certain pressure, the top ITO conductive layer deforms and contacts the bottom ITO layer, forming an equivalent circuit as shown in Figure 2.

Figure 1. LCD touch screen structure

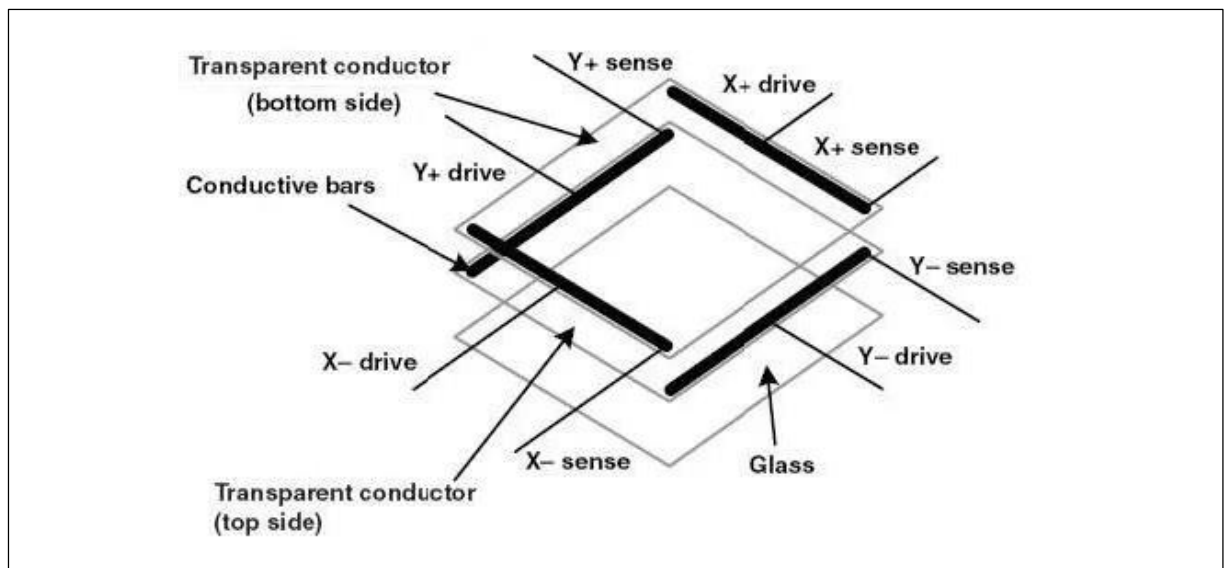
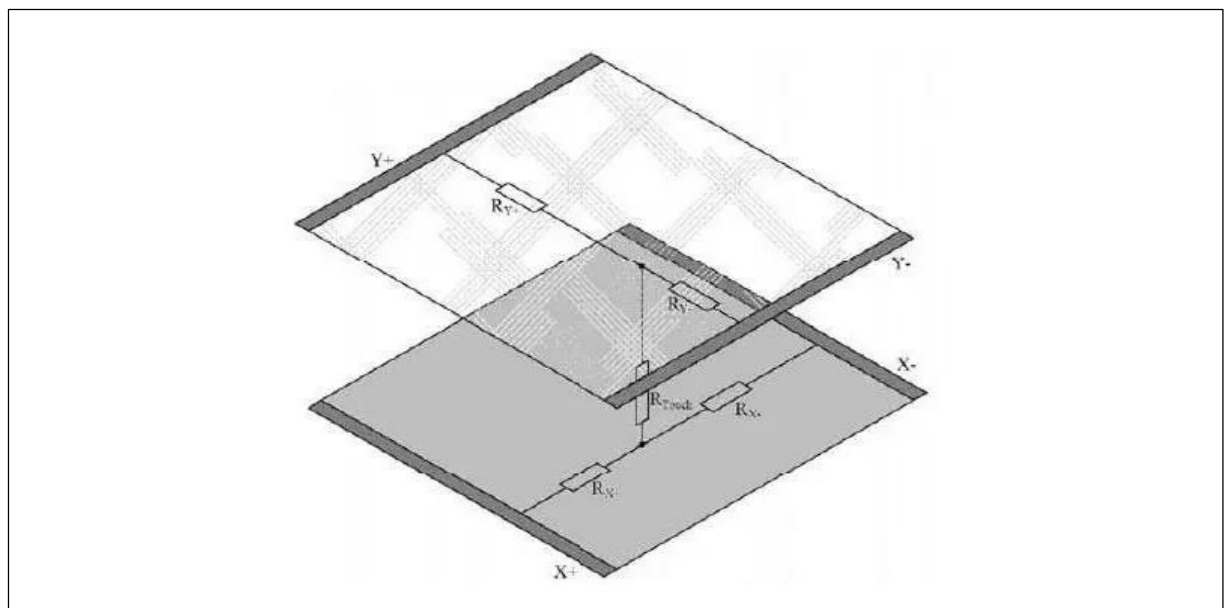


Figure 2. LCD touch screen equivalent circuit



Calculate the X and Y coordinates of the contact point as below:

- 1) Y coordinate: Apply a drive voltage ( $V_{drive}$ ) to the Y+ anode and ground the Y- cathode, and use the X+ anode as the leading-out end to measure the voltage of the contact point. Because the ITO layers are uniformly conductive, the ratio of the contact point voltage to  $V_{drive}$  is equal to the ratio of Y coordinate to the screen height.
- 2) X coordinate: Apply a drive voltage ( $V_{drive}$ ) to the X+ anode and ground the X- cathode, and use the Y+ anode as the leading-out end to measure the voltage of the contact point. Because the ITO layers are uniformly conductive, the ratio of the contact point voltage to  $V_{drive}$  is equal to the ratio of X coordinate to the screen width.

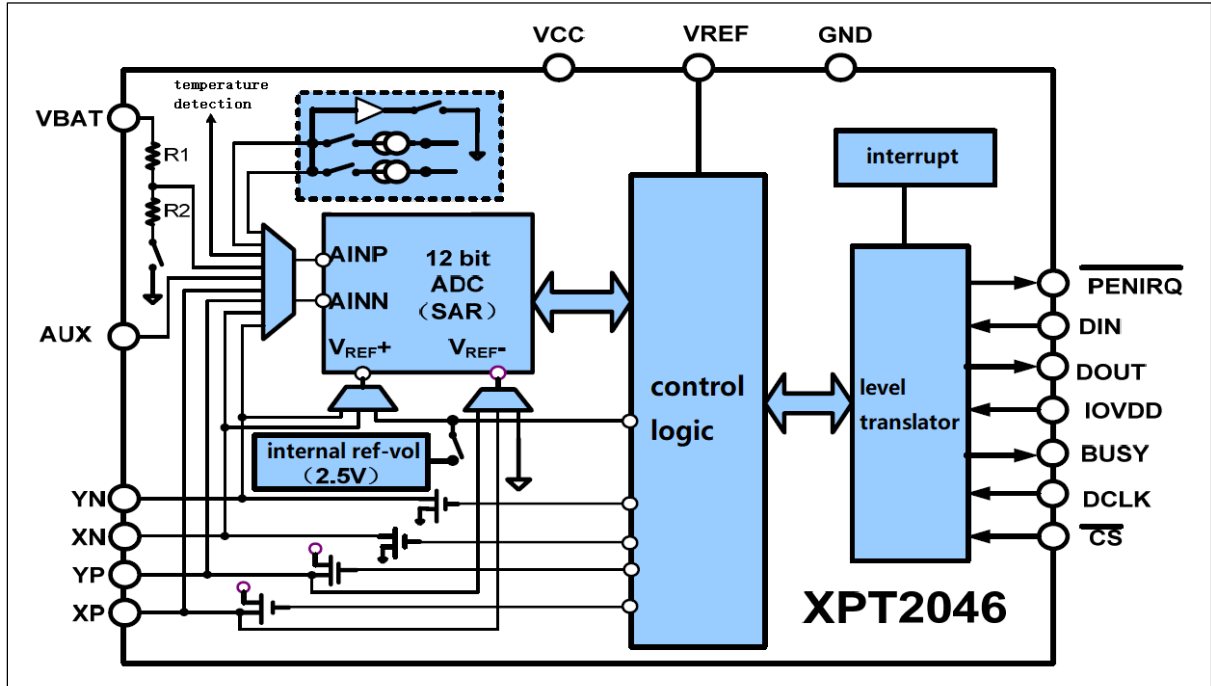
Generally, the measured voltage is converted to a digital signal by ADC, which is then processed as a coordinate to determine the actual position of the contact point. The X and Y coordinates of the contact point on the 4-wire resistive touch screen can be calculated as abovementioned; furthermore, the pressure at the contact point can also be measured based on the resistance that is generated when the top ITO layer is pressed and contacts the bottom ITO layer. The greater the pressure, the greater the contact and the lower the resistance. The pressure can be quantified by measuring the resistance value.

There is always a dedicated MCU for touch screen control in actual applications to implement switching of electrode voltage and collection of voltage values (ADC data) at the contact point. In this application note, the XPT2046 and ILI9341 are used as touch screen controller and display driver, respectively.

## 1.1 Touch sensor

The XPT2046 is a 4-wire resistive touch screen controller that features sample-and-hold, analog-to-digital conversion, and serial data output capabilities. It incorporates a 2.5 V internal reference voltage source, temperature detection circuit and an external clock. Its operation is maintained from a single supply of 2.7 V to 5.5 V. The value of the reference voltage directly sets the input range of the ADC. The reference voltage can be an internal reference voltage, or an external low-impedance source between 1 V and +VCC. Analog signals including X/Y/Z coordinates, VBAT, Temp and AUX are input to the ADC via a controller register. The ADC can be configured as single-ended mode for VBAT, Temp and AUX, or differential mode for touch screen applications to effectively eliminate the measurement errors caused by the parasitic resistance of the driver switch and external interference, and improve the conversion accuracy.

Figure 3. XPT2046 block diagram



## 1.2 LCD display

The LCD is a 240\*320 RGB display controlled by ILI9341. ILI9341 supports parallel and serial data buses. In this application note, the serial interface (SPI) is used for data transfer. The ILI9341 can specify a moving picture area in the internal GRAM by the window address function. The specified window area can be updated selectively, so that moving picture can be displayed simultaneously independent of still picture area. ILI9341 supports full color, 8-color display mode and sleep mode for precise power control by software, and these features make the ILI9341 an ideal LCD driver for medium or small size portable products such as smart phones, MP3 and PMP.

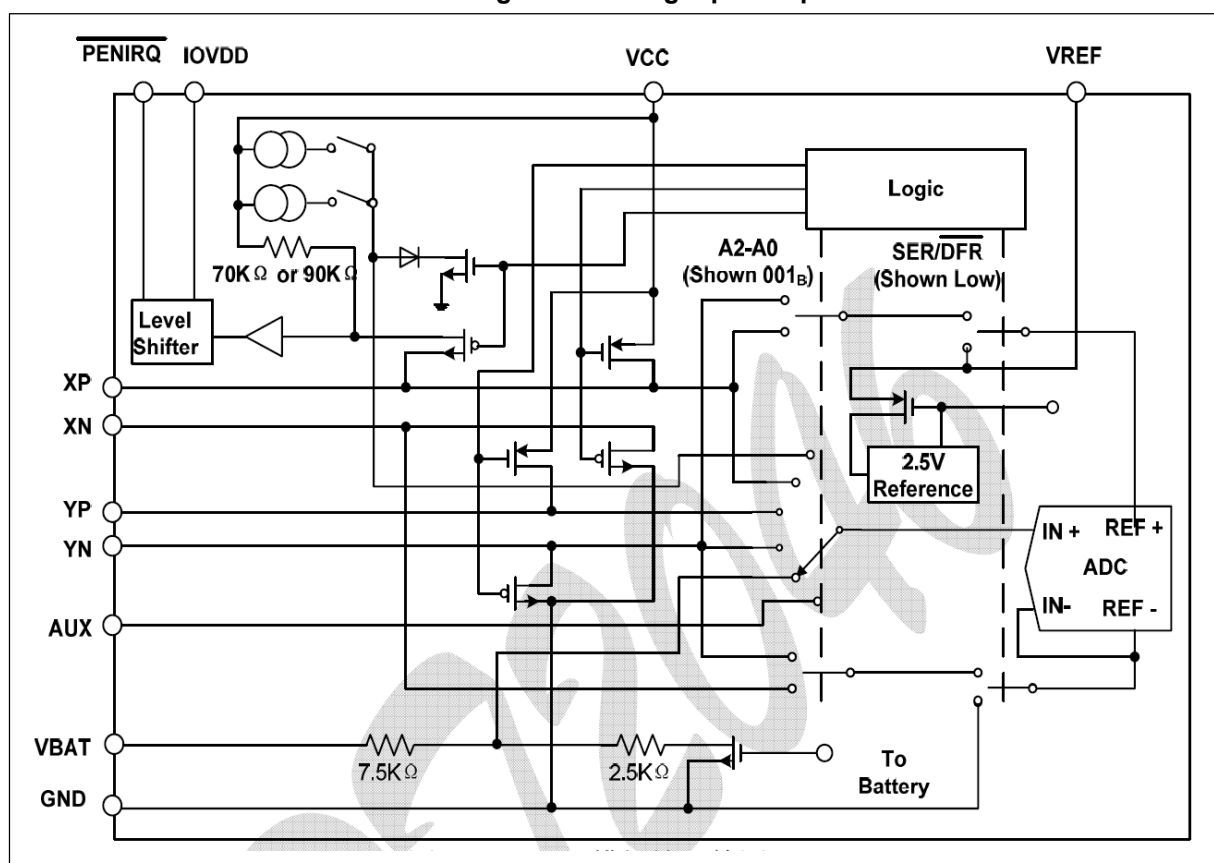
## 2 Operating principles

This section mainly introduces the XPT2046 operating principles.

### 2.1 Analog input

Figure 4 shows the analog differential input and differential reference voltage of multiplexer and ADC in XPT2046.

Figure 4. Analog input loop

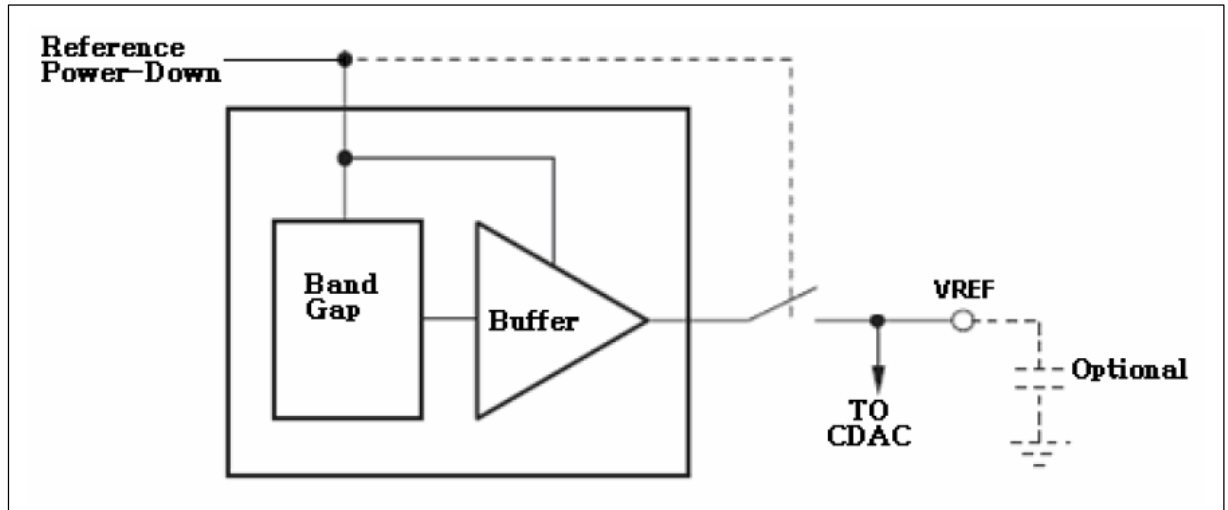


### 2.2 Internal reference voltage

The XPT2046 internal 2.5 V reference voltage source can be turned on or off by setting the control bit PD1. Generally, the internal reference voltage only used for measuring Vbatt, Temp and AUX in single-ended mode. With differential mode, the touch screen can get the best performance. To be compatible with the ADS7843, the internal reference voltage source of XPT2046 must be forcibly turned off by setting PD1=0 after power-on.



Figure 5. Internal voltage source



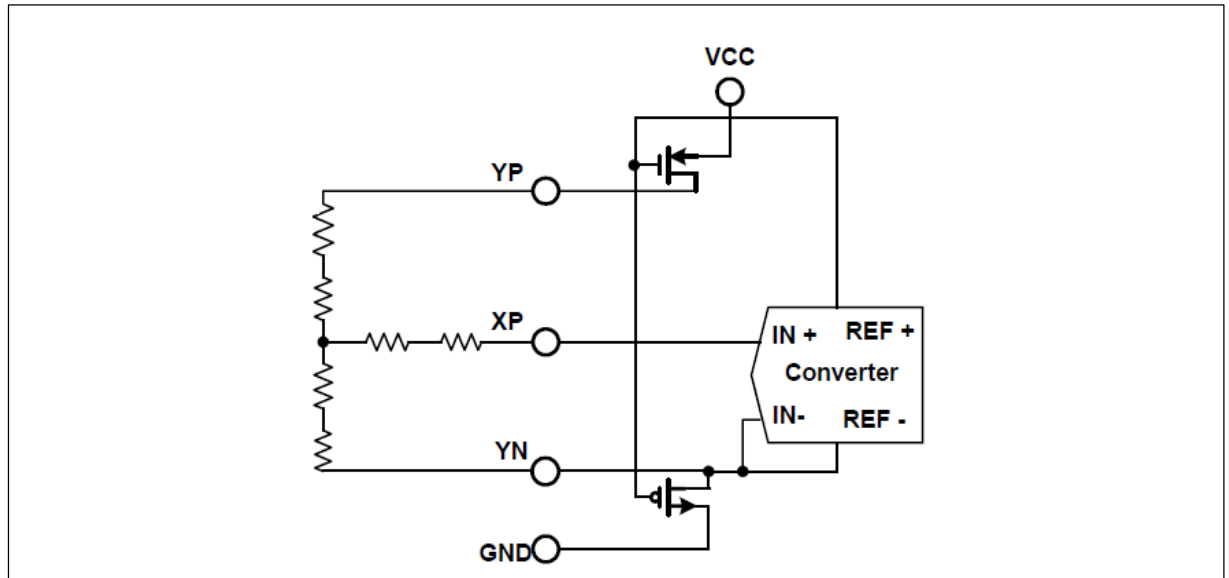
## 2.3 External reference voltage input

The voltage difference ( $V_{REF}$ ) between +REF and -REF (as shown in Figure 3) determines the voltage range of analog input. The reference voltage input range of XPT2046 is 1 V to VCC. The lower the reference voltage, the lower the analog voltage represented by each digit of the ADC output binary data. For 12-bit ADC, the analog voltage represented by the lowest bit of the result is  $V_{REF}/4096$  (and so forth). The lower the reference voltage, the greater the error caused by interference; therefore, a low-noise and low-fluctuation reference voltage source should be used. To minimize interference, the input signal noise should be reduced as much as possible when designing the circuit board, thus to guarantee the conversion accuracy.

## 2.4 Differential mode

As mentioned before, when the XPT2046 is used for touch screen applications, it can be configured as differential mode, so that the +REF and -REF inputs can be directly connected to YP and YN, respectively, thus to eliminate coordinate measurement errors caused by the on-resistance of the driver switch. However, in differential mode, the driver switch needs to be switched on during both sampling and conversion, which cause greater power consumption than the single-ended mode. When SER/DFR are set to low, the XPT2046 is in differential mode, as shown in Figure 6 below.

Figure 6. Reference voltage source in differential mode

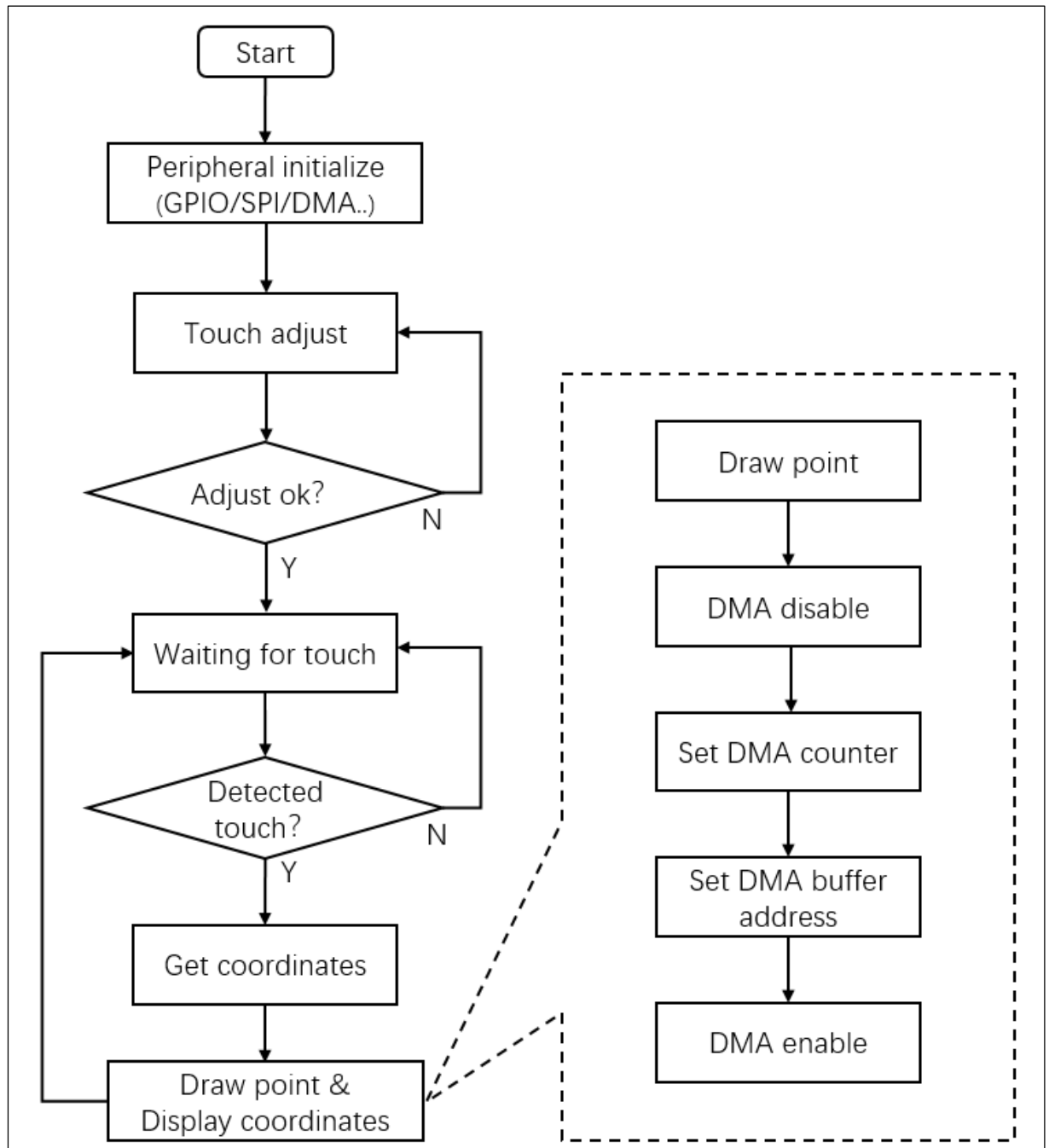


## 2.5 Software operation

This section mainly introduces the touch screen driver software code flow.

First, power on and then initialize related peripherals, including CRM, GPIO, SPI and DMA, and then perform touch calibration by calling the “touch\_adjust()” function. After successful calibration, a set of calibration parameters will be displayed for the measurement and calculation of contact point coordinates. Then, perform touch screen test, and the contact point and its coordinates will be drawn on the screen during the test. In addition, a CLEAR button is set, and users can click on this button to clear all the contents including coordinates on the screen. In order to improve the efficiency of data transmission, the clear screen and point data in this demo are transmitted through DMA-SPI. The software operation flow is shown in Figure 7 below.

Figure 7. Software operation flow



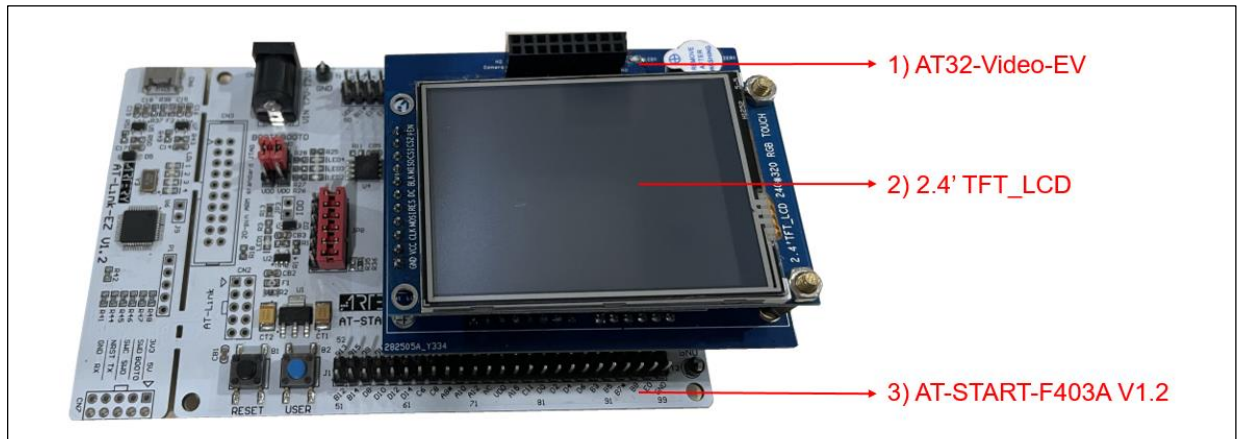
*Note: It is necessary to complete touch calibration before follow-up operations.*

### 3 How to use LCD touch screen

#### 3.1 Hardware

- 1) AT32-Video-EV
- 2) 2.4' TFT\_LCD
- 3) AT-START-F403A V1.2 evaluation board

Figure 8. AT-START-F403A V1.2 evaluation board and LCD touch



Note:

1. This demo is based on the AT32F403A series. If users want to use this demo for other AT32 MCU series, please modify related configurations correspondingly.
2. Power supply: Direct power supply or power supply by using a USB cable (do not use Link power supply separately).

#### 3.2 Software

AN0154\_LCD\_Touch\_Sourcecode: LCD touch screen test mode, stored in  
Sourcecode\utilities\mdk\_v5

#### 3.3 Main codes

- 1) Touch point coordinate get function: It is used to get the coordinate of the touch point. The main codes are shown below.

```
uint8_t touch_coor_read_twice(uint16_t *x, uint16_t *y)
{
    uint16_t x1, y1;
    uint16_t x2, y2;
    uint8_t flag;
    flag = touch_coor_read(&x1, &y1);

    if(flag == 0)
    {
        return(0);
    }
}
```

```

flag = touch_coor_read(&x2, &y2);

if(flag == 0)
{
    return(0);
}

if(((x2 <= x1 && x1 < x2 + ERR_RANGE) || (x1 <= x2 && x2 < x1 + ERR_RANGE))
    && ((y2 <= y1 && y1 < y2 + ERR_RANGE) || (y1 <= y2 && y2 < y1 + ERR_RANGE)))
{
    *x = (x1 + x2) >> 1;
    *y = (y1 + y2) >> 1;
    return 1;
}
else
{
    return 0;
}
}

```

- 2) Touch calibration function: It is used to get the calibration parameters. The main codes are shown below.

```

void touch_adjust(void)
{
    float vx1, vx2, vy1, vy2;
    uint16_t chx1, chx2, chy1, chy2;
    uint16_t lx, ly;
    struct tp_pixu32_ p[4];
    uint8_t cnt = 0;
    cnt = 0;
    POINT_COLOR = RED;
    lcd_clear(WHITE);
    touch_adjust_point_draw(TOUCH_OFFSET, TOUCH_OFFSET);

    while(1)
    {
        if(PEN == 0)
        {
            if(touch_precise_coor_read())
            {
                p[cnt].x = tp_pixad.x;
                p[cnt].y = tp_pixad.y;
                cnt++;
            }

            switch(cnt)
            {
                case 1:
                    lcd_clear(WHITE);

```

```

while(!PEN)
{
}

touch_adjust_point_draw(LCD_WIDTH - TOUCH_OFFSET - 1, TOUCH_OFFSET);
break;

case 2:
    lcd_clear(WHITE);

    while(!PEN)
    {
    }

    touch_adjust_point_draw(TOUCH_OFFSET, LCD_HEIGHT - TOUCH_OFFSET - 1);
    break;

case 3:
    lcd_clear(WHITE);

    while(!PEN)
    {
    }

    touch_adjust_point_draw(LCD_WIDTH - TOUCH_OFFSET - 1, LCD_HEIGHT -
    TOUCH_OFFSET - 1);
    break;

case 4:
    lcd_clear(WHITE);

    while(!PEN)
    {
    }

    vx1 = p[1].x > p[0].x ? (p[1].x - p[0].x + 1) * 1000 / (LCD_WIDTH - TOUCH_OFFSET -
    TOUCH_OFFSET) : (p[0].x - p[1].x - 1) * 1000 / (LCD_WIDTH - TOUCH_OFFSET -
    TOUCH_OFFSET);
    chx1 = p[1].x > p[0].x ? p[0].x - (vx1 * TOUCH_OFFSET) / 1000 : p[0].x + (vx1 *
    TOUCH_OFFSET) / 1000;
    vy1 = p[2].y > p[0].y ? (p[2].y - p[0].y - 1) * 1000 / (LCD_HEIGHT - TOUCH_OFFSET -
    TOUCH_OFFSET) : (p[0].y - p[2].y - 1) * 1000 / (LCD_HEIGHT - TOUCH_OFFSET -
    TOUCH_OFFSET);
    chy1 = p[2].y > p[0].y ? p[0].y - (vy1 * TOUCH_OFFSET) / 1000 : p[0].y + (vy1 *
    TOUCH_OFFSET) / 1000;

    vx2 = p[3].x > p[2].x ? (p[3].x - p[2].x + 1) * 1000 / (LCD_WIDTH - TOUCH_OFFSET -
    TOUCH_OFFSET) : (p[2].x - p[3].x - 1) * 1000 / (LCD_WIDTH - TOUCH_OFFSET -
    TOUCH_OFFSET);
    chx2 = p[3].x > p[2].x ? p[2].x - (vx2 * TOUCH_OFFSET) / 1000 : p[2].x + (vx2 *
    TOUCH_OFFSET) / 1000;
    vy2 = p[3].y > p[1].y ? (p[3].y - p[1].y - 1) * 1000 / (LCD_HEIGHT - TOUCH_OFFSET -
    TOUCH_OFFSET) : (p[1].y - p[3].y - 1) * 1000 / (LCD_HEIGHT - TOUCH_OFFSET -

```

```

TOUCH_OFFSET);
    chy2 = p[3].y > p[1].y ? p[1].y - (vy2 * TOUCH_OFFSET) / 1000 : p[1].y + (vy2 *
TOUCH_OFFSET) / 1000;

    if((vx1 > vx2 && vx1 > vx2 + TOUCH_ADJUST) || (vx1 < vx2 && vx1 < vx2 -
TOUCH_ADJUST) || (vy1 > vy2 && vy1 > vy2 + TOUCH_ADJUST) || (vy1 < vy2 && vy1 < vy2 -
TOUCH_ADJUST))
    {
        cnt = 0;
        lcd_clear(WHITE);
        touch_adjust_point_draw(TOUCH_OFFSET, TOUCH_OFFSET);
        continue;
    }

    vx = (vx1 + vx2) / 2;
    vy = (vy1 + vy2) / 2;
    chx = (chx1 + chx2) / 2;
    chy = (chy1 + chy2) / 2;
    ...
    }
}
}
}
}

```

- 3) Touch screen test function: It is used to test the touch screen. The main codes are shown below.

```

void touch_test(void)
{
    double t = 0;

    while(1)
    {
        if(PEN == 0)
        {
            t = 0;

            if(coor_convert())
            {
                if(tp_pixlcd.x > 200 && tp_pixlcd.y > 300)
                {
                    lcd_clear(WHITE);
                    POINT_COLOR = WHITE;
                    BACK_COLOR = RED;
                    lcd_showstring(200,300,200,16,16,"Clear");
                    POINT_COLOR = RED;
                    BACK_COLOR = WHITE;
                    lcd_showstring(10, 300, 200, 16, 16, "X:");
                    lcd_shownum(30, 300, 0, 3, 16);
                    lcd_showstring(100, 300, 200, 16, 16, "Y:");
                    lcd_shownum(120, 300, 0, 3, 16);
                }
            }
        }
    }
}

```

```

    }
    else
    {
        lcd_showstring(10, 300, 200, 16, 16, "X:");
        lcd_shownum(30, 300, (u32)tp_pixlcd.x, 3, 16);
        lcd_showstring(100, 300, 200, 16, 16, "Y:");
        lcd_shownum(120, 300, (u32)tp_pixlcd.y, 3, 16);
        lcd_drawpoint_big(tp_pixlcd.x, tp_pixlcd.y);
    }
}
}
else
{
    t++;

    if(t > 65000)
    {
        return;
    }
}
}
}
}

```

- 4) LCD touch screen clear function: It is used to clear all contents on the screen. The main codes are shown below.

```

void lcd_clear(u16 color)
{
    uint32_t i;
    u32 index = 0;
    u32 totalpoint = lcddev.width;
    totalpoint *= lcddev.height;
    for(i = 0; i < LCD_DMA_SIZE; i++)
    {
        touch_point_color[i] = color;
    }

    blockwrite(0, lcddev.width, 0, lcddev.height);

    for(index = 0; index < 4; index++)
    {
        LCD_DC_SET;
        LCD_SPI_MASTER_Tx_DMA_Channel->ctrl_bit.chen = FALSE;
        LCD_SPI_MASTER_Tx_DMA_Channel->dtcnt_bit.cnt = LCD_DMA_SIZE*2;
        LCD_SPI_MASTER_Tx_DMA_Channel->maddr = (uint32_t)touch_point_color;
        LCD_SPI_MASTER_Tx_DMA_Channel->ctrl_bit.chen = TRUE;
        while(dma_flag_get(LCD_SPI_MASTER_Tx_DMA_FLAG) != 1);
        dma_flag_clear(LCD_SPI_MASTER_Tx_DMA_FLAG);
    }
}

```



- 5) LCD touch screen point draw function: It is used to display the touch point coordinate. The main codes are shown below.

```
void lcd_draw_point_big(u16 x, u16 y)
{
    u16 i;

    for(i = 0; i < TOUCH_POINT_SIZE; i++)
    {
        lcd_set_cursor(x - 1, y - 1 + i);
        lcd_writeram_prepare();

        LCD_DC_SET;
        LCD_SPI_MASTER_Tx_DMA_Channel->ctrl_bit.chen = FALSE;
        LCD_SPI_MASTER_Tx_DMA_Channel->dtcnt_bit.cnt = TOUCH_POINT_SIZE*2;
        LCD_SPI_MASTER_Tx_DMA_Channel->maddr = (uint32_t)touch_point_color;
        LCD_SPI_MASTER_Tx_DMA_Channel->ctrl_bit.chen = TRUE;
        while(dma_flag_get(LCD_SPI_MASTER_Tx_DMA_FLAG) != 1);
        dma_flag_clear(LCD_SPI_MASTER_Tx_DMA_FLAG);
    }
}
```

### 3.4 LCD Touch demo

Perform the following steps to use LCD Touch demo:

- 1) Download and compile LCD touch screen demo test code;
- 2) Enter touch screen adjust interface, and click four calibration points in order, as shown in Figure 9.
- 3) After the completion of calibration, LCD displays the calibration information (including VX, VY, CHX and CHY), as shown in Figure 9.
- 4) Click the screen to enter the touch screen test interface. At this point, touch points are drawn on the LCD touch screen, and the coordinates of the touch points are displayed in a real-time manner. Click "Clear" button to clear the touch screen, as shown 11.

Figure 9. Touch screen adjust interface



Figure 10. Touch screen adjust information

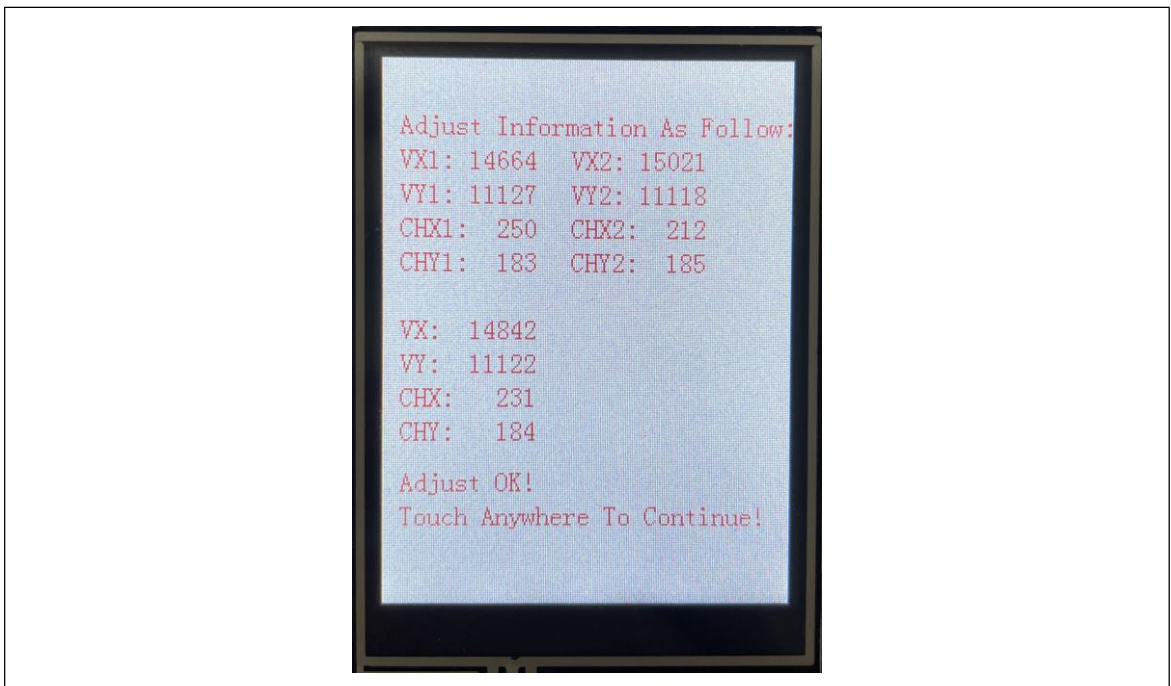


Figure 11. Touch screen test interface



## 4 Revision history

Table 1. Document revision history

Date	Version	Revision note
2022.10.12	2.0.0	Initial release.
2022.11.08	2.0.1	Added software operation section and some main codes.

## IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services, and ARTERY assumes no liability whatsoever relating to the choice, selection or use of the ARTERY products and services described herein

No license, express or implied, to any intellectual property rights is granted under this document. If any part of this document deals with any third party products or services, it shall not be deemed a license granted by ARTERY for the use of such third party products or services, or any intellectual property contained therein, or considered as a warranty regarding the use in any manner of such third party products or services or any intellectual property contained therein.

Unless otherwise specified in ARTERY's terms and conditions of sale, ARTERY provides no warranties, express or implied, regarding the use and/or sale of ARTERY products, including but not limited to any implied warranties of merchantability, fitness for a particular purpose (and their equivalents under the laws of any jurisdiction), or infringement on any patent, copyright or other intellectual property right.

Purchasers hereby agree that ARTERY's products are not designed or authorized for use in: (A) any application with special requirements of safety such as life support and active implantable device, or system with functional safety requirements; (B) any aircraft application; (C) any aerospace application or environment; (D) any weapon application, and/or (E) or other uses where the failure of the device or product could result in personal injury, death, property damage. Purchasers' unauthorized use of them in the aforementioned applications, even if with a written notice, is solely at purchasers' risk, and Purchasers are solely responsible for meeting all legal and regulatory requirements in such use.

Resale of ARTERY products with provisions different from the statements and/or technical characteristics stated in this document shall immediately void any warranty grant by ARTERY for ARTERY's products or services described herein and shall not create or expand any liability of ARTERY in any manner whatsoever.

© 2022 ARTERY Technology – All Rights Reserved