

AT32 LCD Touch Driven By SPI

前言

本应用笔记的目的是提供在AT32微控制器上实现SPI驱动触摸屏应用程序的一般方法。

支持型号列表：

支持型号	AT32F403Axx
	AT32F407xx

目录

1	触摸屏介绍	5
1.1	触摸感应器	6
1.2	LCD 显示器	6
2	触摸屏工作原理.....	7
2.1	模拟输入.....	7
2.2	内部参考电压.....	7
2.3	外部参考电压输入.....	8
2.4	差分工作模式.....	8
2.5	软件流程.....	9
3	触摸屏快速使用方法	10
3.1	硬件资源.....	10
3.2	软件资源.....	10
3.3	关键代码.....	10
3.4	LCD Touch demo 使用.....	15
4	版本历史	17

表目录

表 1. 文档版本历史	17
-------------------	----

图目录

图 1. 触摸屏结构图	5
图 2. 触摸屏等效电路	5
图 3. XPT2046 原理框图	6
图 4. 模拟输入简图	7
图 5. 内部电压源示意图	8
图 6. 差分参考源工作模式简图	8
图 7. 软件流程图	9
图 8. 触摸屏硬件资源图	10
图 9. 触摸屏校准界面	15
图 10. 触摸屏校准信息	16
图 11. 触摸屏测试界面	16

1 触摸屏介绍

对于四线电阻式触摸屏的结构如下图1，在玻璃或丙烯酸基板上覆盖有两层透明，均匀导电的ITO层，分别做为X电极和Y电极，它们之间由均匀排列的透明格点分开绝缘。其中下层的ITO与玻璃基板附着，上层的ITO附着在PET薄膜上。X电极和Y电极的正负端由“导电条”（图中黑色条形部分）分别从两端引出，且X电极和Y电极导电条的位置相互垂直。引出端X-，X+，Y-，Y+一共四条线，这就是四线电阻式触摸屏名称的由来。当有物体接触触摸屏表面并施以一定的压力时，上层的ITO导电层发生形变与下层ITO发生接触，该结构可以等效为相应的电路，如下图2。

图 1. 触摸屏结构图

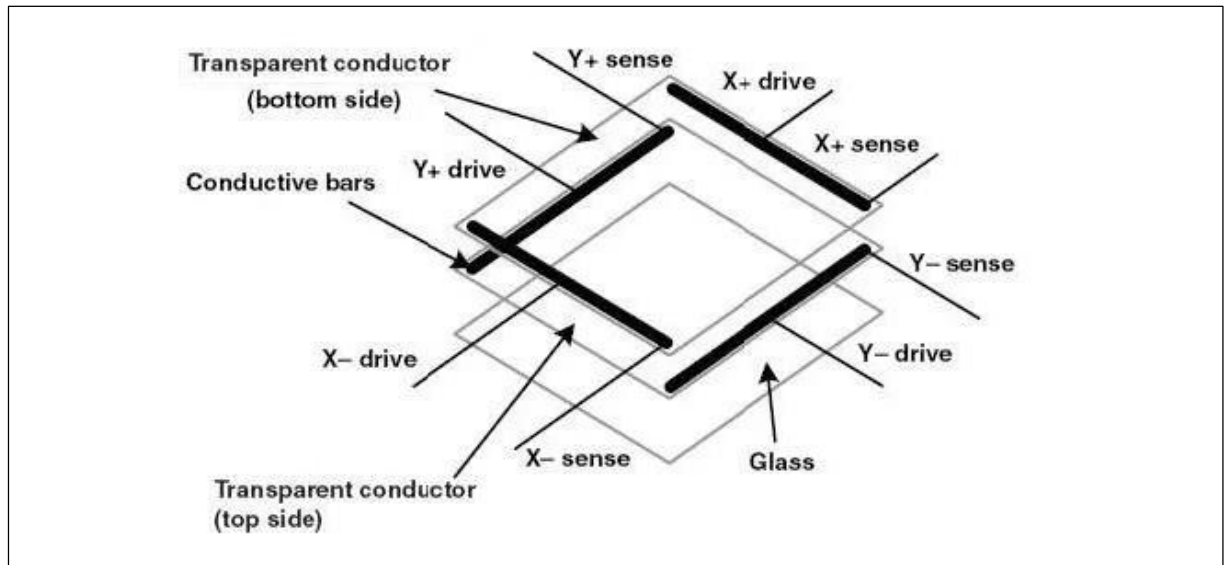
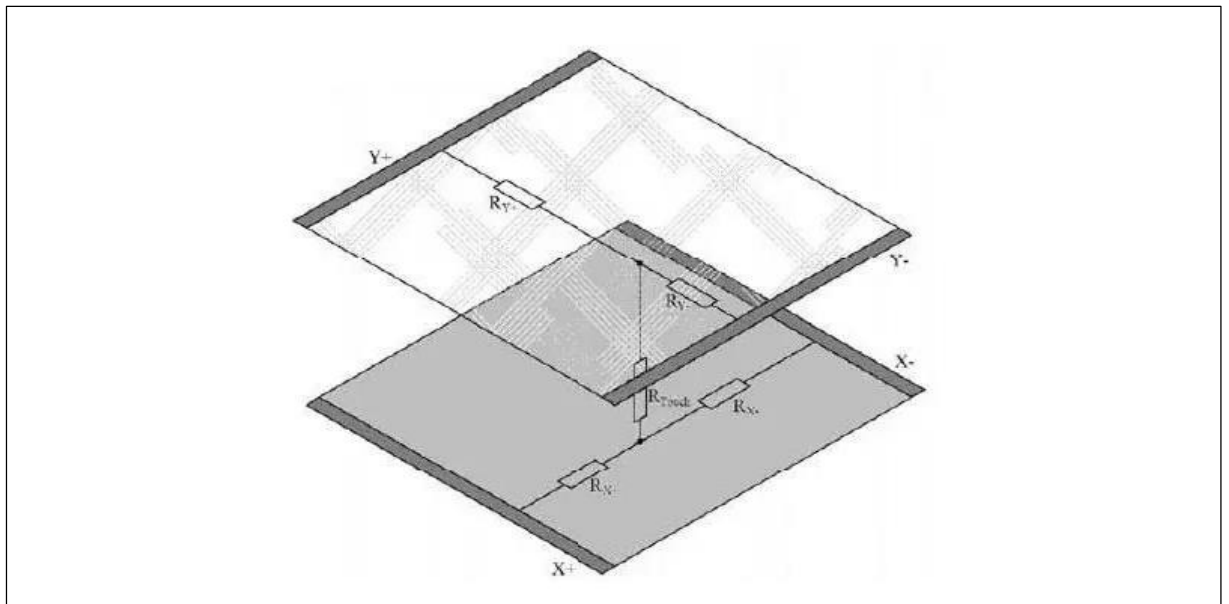


图 2. 触摸屏等效电路



计算触点的X，Y坐标分为如下两步：

- 1) 计算Y坐标，在Y+电极施加驱动电压Vdrive，Y-电极接地，X+做为引出端测量得到接触点的电压，由于ITO层均匀导电，触点电压与Vdrive电压之比等于触点Y坐标与屏高度之比。
- 2) 计算X坐标，在X+电极施加驱动电压Vdrive，X-电极接地，Y+做为引出端测量得到接触点的电压，由于ITO层均匀导电，触点电压与Vdrive电压之比等于触点X坐标与屏宽度之比。

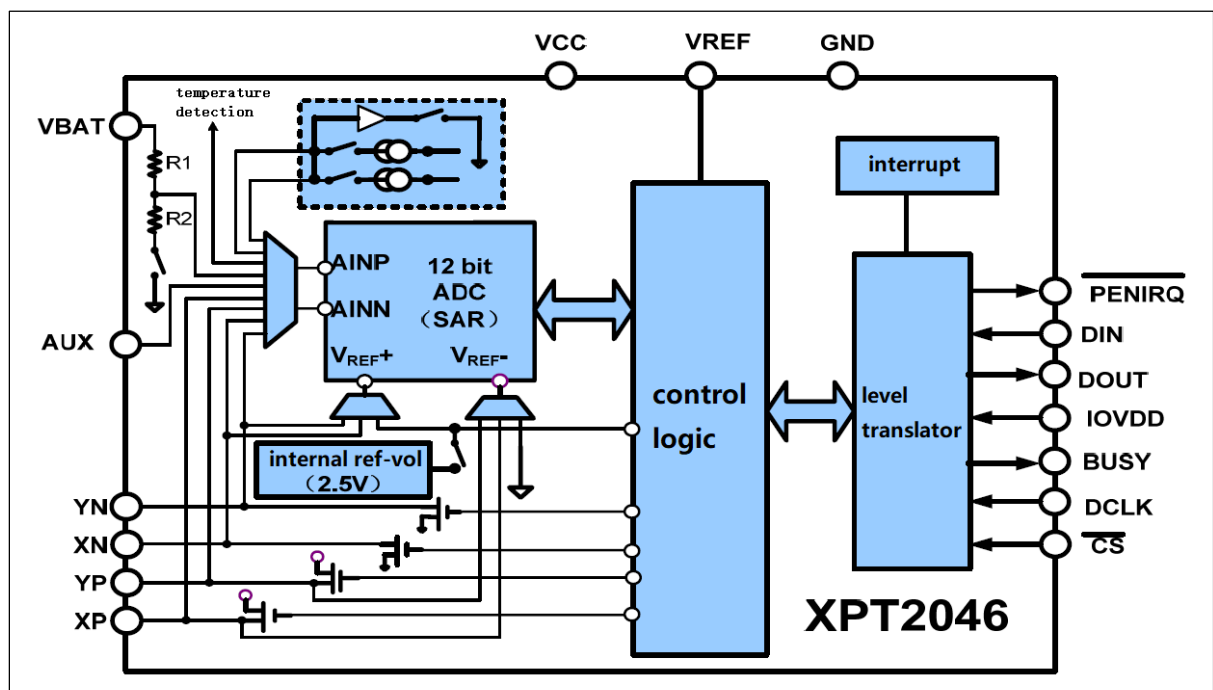
测得的电压通常由 ADC 转化为数字信号，再进行简单处理就可以做为坐标判断触点的实际位置。四线电阻式触摸屏除了可以得到触点的 X/Y 坐标，还可以测得触点的压力，这是因为 top layer 施压后，上下层 ITO 发生接触，在触点上实际是有电阻存在的。压力越大，接触越充分，电阻就越小，通过测量这个电阻的大小可以量化压力大小。

通常在触摸屏应用中对于触摸屏控制有专门的控制芯片，主要就是为了完成两个任务：其一，完成电极电压的切换；其二，采集接触点处的电压值（ADC 数据）。本案例中触摸屏使用的触摸感应驱动芯片为 XPT2046，显示器驱动芯片为 ILI9341，下面将分别做介绍。

1.1 触摸感应器

触摸感应器使用的驱动芯片为 XPT2046，其包含了采样/保持、模数转换、串口数据输出等功能。同时芯片集成有一个 2.5V 的内部参考电压源、温度检测电路，工作时使用外部时钟。XPT2046 可以单电源供电，电源电压范围为 2.7V~5.5V。参考电压值直接决定 ADC 的输入范围，参考电压可以使用内部参考电压，也可以从外部直接输入 1V~VCC 范围内的参考电压（要求外部参考电压源输出阻抗低）。X、Y、Z、VBAT、Temp 和 AUX 模拟信号经过片内的控制寄存器选择后进入 ADC，ADC 可以配置为单端或差分模式。选择 VBAT、Temp 和 AUX 时可以配置为单端模式；作为触摸屏应用时，可以配置为差分模式，这可有效消除由于驱动开关的寄生电阻及外部的干扰带来的测量误差，提高转换准确度。

图 3. XPT2046 原理框图



1.2 LCD 显示器

LCD 显示器为一块 240*320 的 RGB 屏幕，使用驱动芯片为 ILI9341。ILI9341 能够支持并行和串行数据总线，此案例中我们使用串行总线接口（SPI）来进行数据传输。ILI9341 驱动器能够通过窗口地址函数在内部 GRAM 中指定动态图像的区域，并且可选择地更新此窗口区域，这样就可以在独立于静态图像区域的同时显示动态图像。ILI9341 支持全彩色，8 色显示模式和休眠模式，能够通过软件进行精确的电源控制，使得 ILI9341 能够作为手机、MP3 和 PMP 等便携设备理想的液晶驱动器。

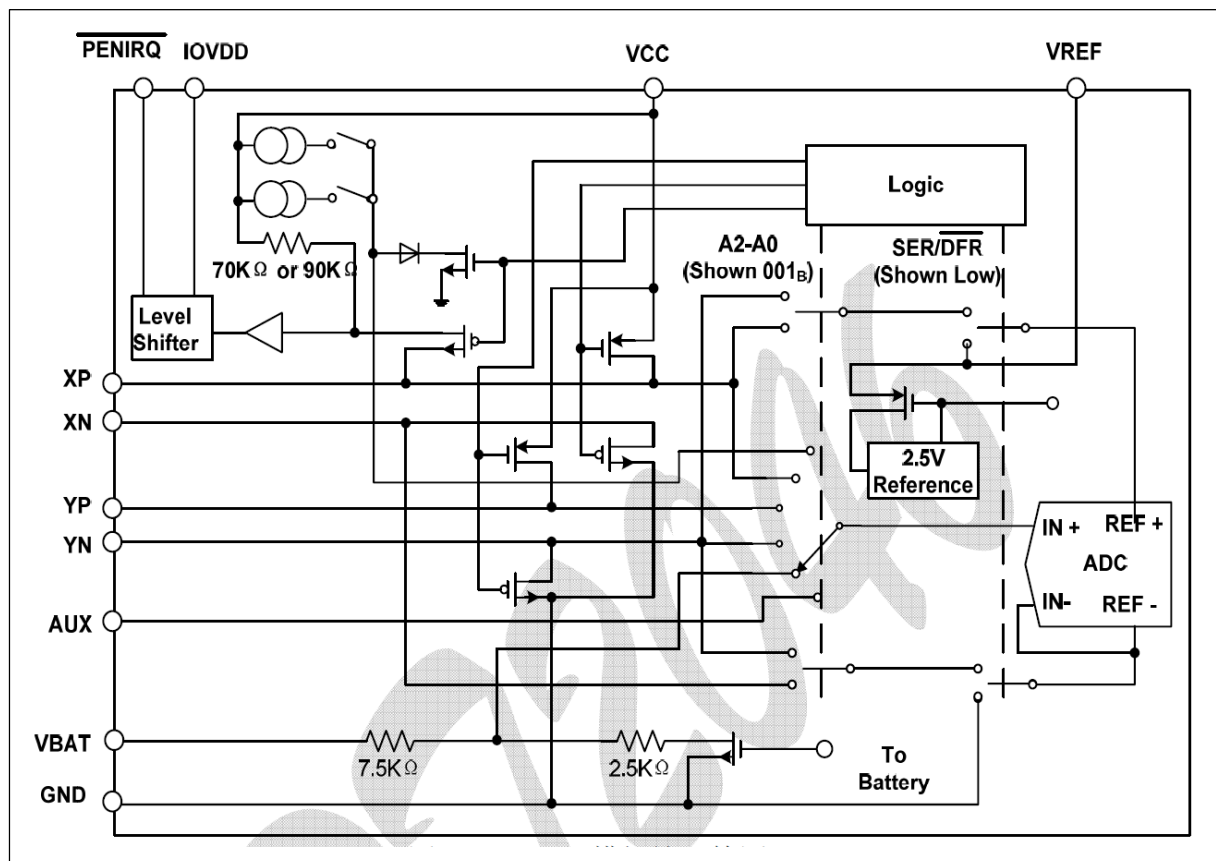
2 触摸屏工作原理

下面将对XPT2046驱动芯片的工作原理进行介绍。

2.1 模拟输入

下图描述了XPT2046片内多路选择器、ADC的模拟差分输入和差分参考电压基准。

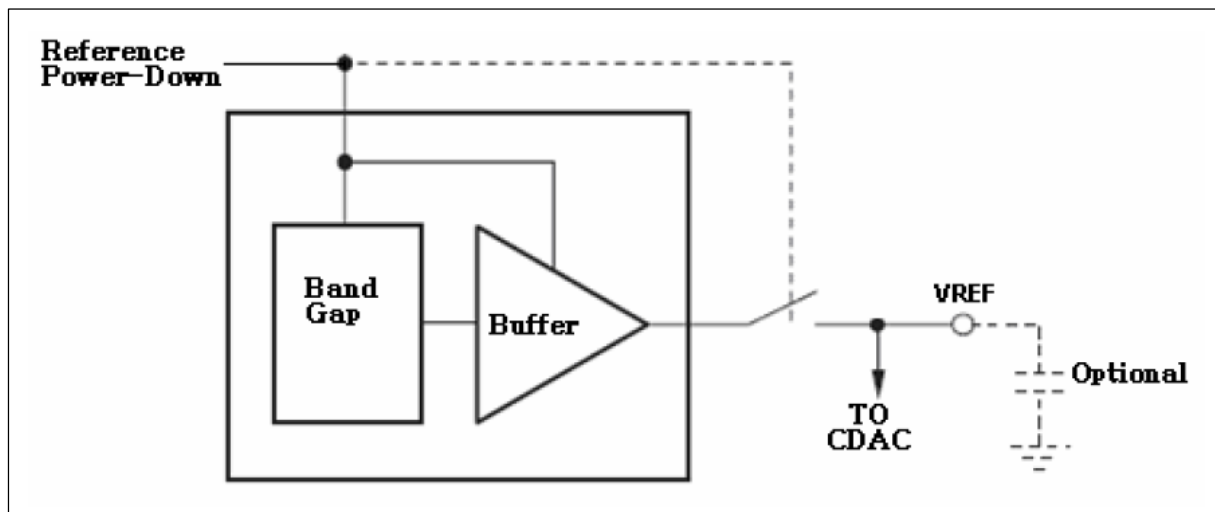
图 4. 模拟输入简图



2.2 内部参考电压

XPT2046的内部2.5V参考电压源可通过控制位PD1进行关闭或者打开。一般地，内部参考电压只用于单端模式下Vbatt、Temp和AUX输入测量。使用差分模式，触摸屏可以获得最佳性能。如果要与ADS7843兼容，XPT2046的内部参考电压源必须强行关闭。因此，上电后要对控制位PD1置0以确保关闭内部参考源。

图 5. 内部电压源示意图



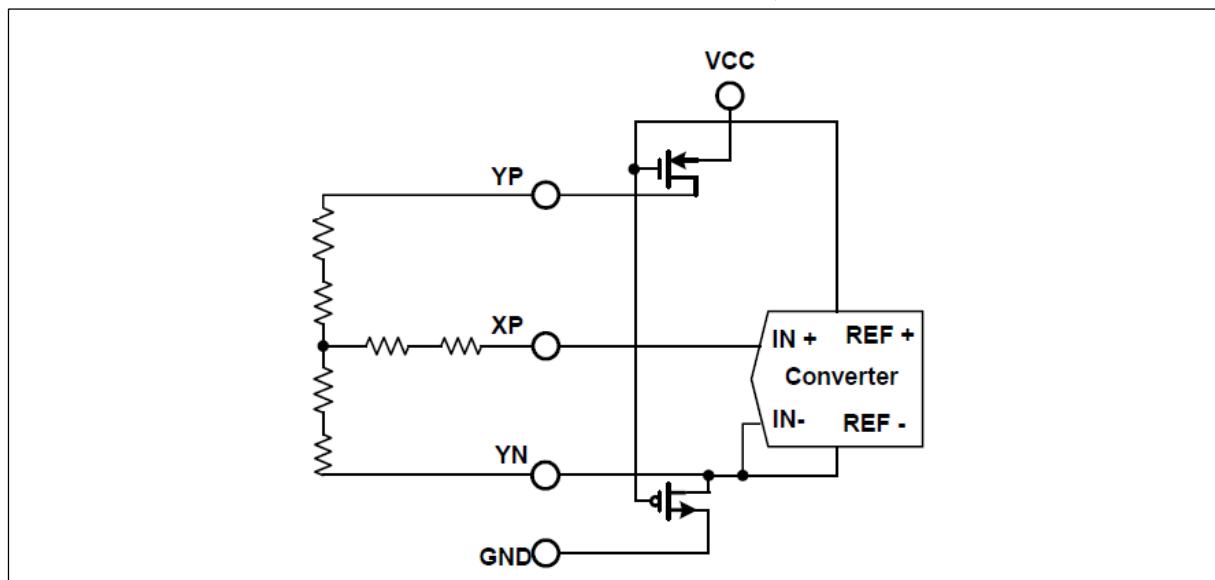
2.3 外部参考电压输入

+REF和-REF（见图3）之间的电压差（下文用VREF表示）决定了模拟输入的电压范围。XPT2046的参考电压输入范围为1V~ VCC。参考电压越低，则ADC输出的二进制数据结果每一个数字位所代表的模拟电压也越低。在12位工作方式下，数据结果的最低位所代表的模拟电压为 $VREF / 4096$ ，其余位依此类推。因此，参考电压越低，干扰引入的误差会越大，此时要求尽可能使用低噪声、低波动的参考电压源；在设计电路板时，尽可能减少干扰，输入的信号噪音也不能太高，否则会直接影响转换精度。

2.4 差分工作模式

如前所述，当触摸感应器XPT2046作为触摸屏应用时，可以配置为差分模式。差分模式的优点是：+REF和-REF的输入分别直接接到YP、YN上，可消除由于驱动开关的导通电阻引入的坐标测量误差。缺点是：无论是采样还是转换过程中，驱动开关都需要接通，相对单端模式而言，功耗会有增加。当SER/DFR置为低电平时，XPT2046为差分工作模式，如下图所示。

图 6. 差分参考源工作模式简图

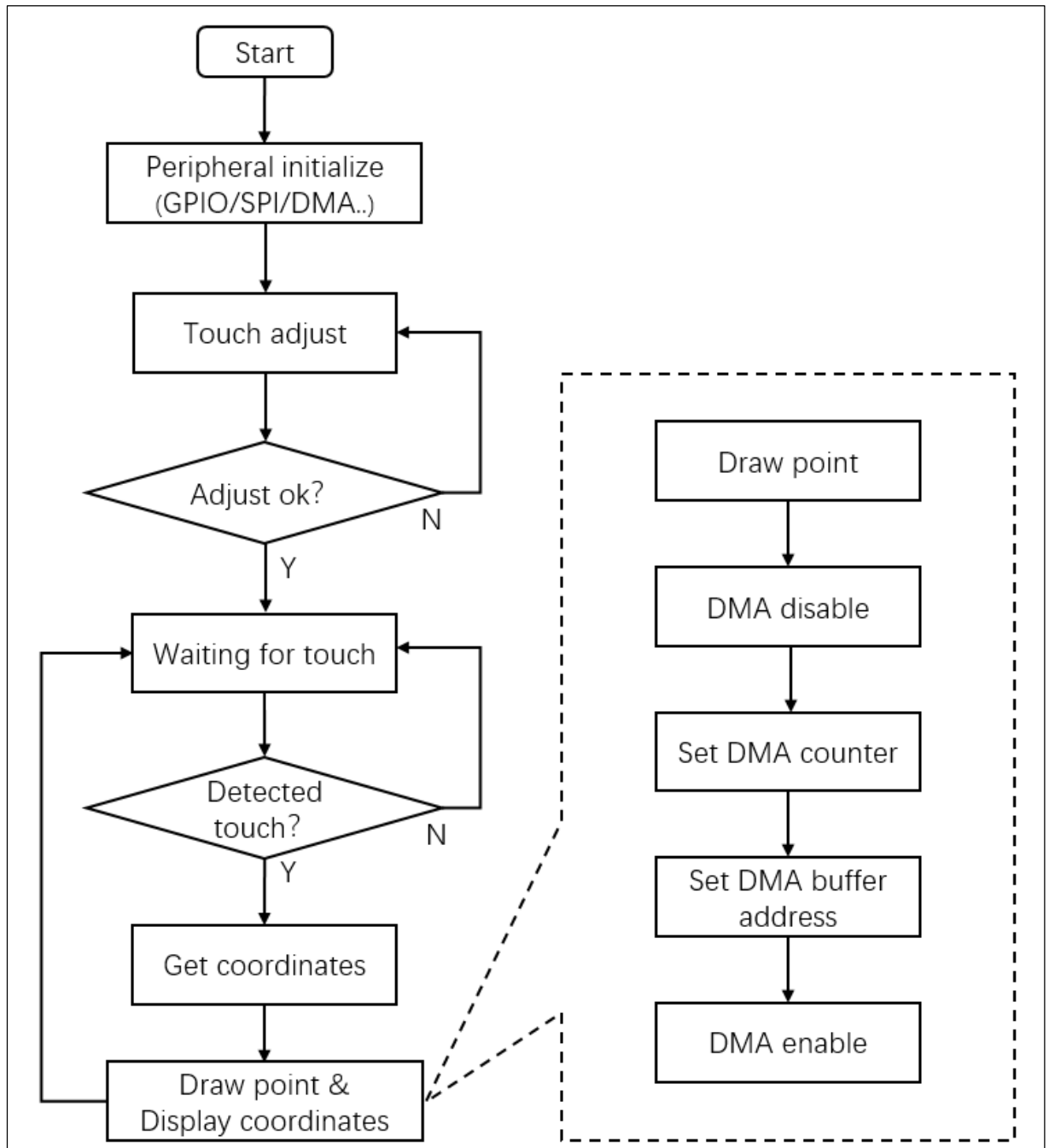


2.5 软件流程

接下来将简单介绍本案例的触摸屏驱动软件代码流程。

首先上电之后需要对所用到的外设进行初始化，包括CRM、GPIO、SPI和DMA等外设。外设初始化完成后即可开始进行触摸校准，通过调用 `touch_adjust()` 函数完成。校准完成后会在屏幕显示一系列的校准参数，用于后续计算触摸坐标使用。随后，即可开始进行触摸测试，测试过程中会在屏幕画出已触摸的坐标点，以及坐标值。还设置了清屏按钮，点击即可清除屏幕已显示内容和坐标值。为了提高数据传输效率，此demo中清屏和画点数据通过DMA-SPI进行传输，软件流程图如下：

图 7. 软件流程图



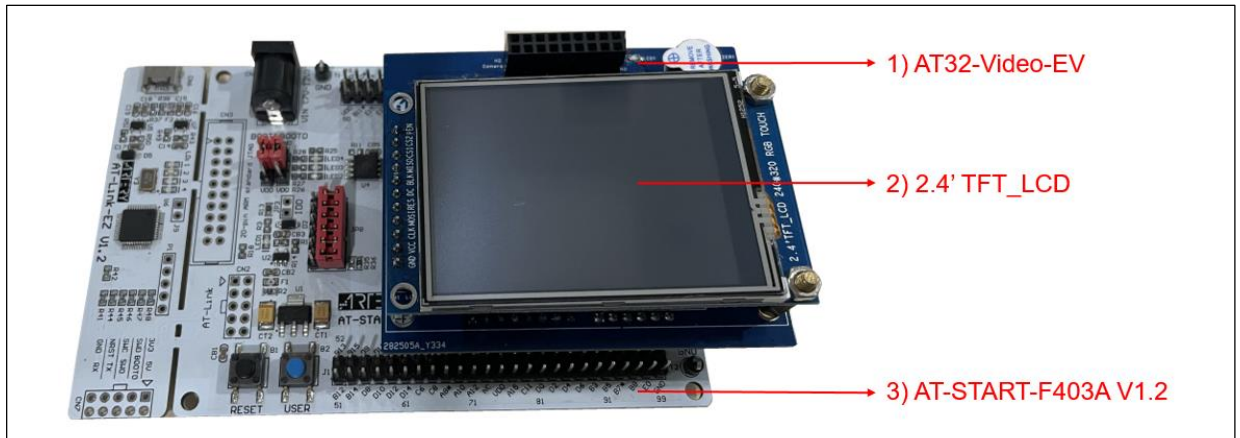
注：由于触摸校准对触摸功能来说是必须完成的，如果未完成校准则无法进行后续的工作。

3 触摸屏快速使用方法

3.1 硬件资源

- 1) AT32-Video-EV
- 2) 2.4寸TFT_LCD
- 3) AT-START-F403A V1.2 实验板

图 8. 触摸屏硬件资源图



Note:

1. 该 demo 是基于 AT32F403A 的硬件条件，若使用者需要在 AT32 其他型号上使用，请修改相应配置即可。
2. 供电部分：使用电源供电，或 USB 线供电（勿使用 Link 单独供电）。

3.2 软件资源

AN0154_LCD_Touch_Sourcecode, 触摸屏案例测试代码，工程路径位于：
Sourcecode\utilities\mdk_v5

3.3 关键代码

- 1) 触摸点坐标获取函数，用于获取触摸坐标值，关键代码如下：

```
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) 触摸校准函数，用于获取校准参数，关键代码如下：

```
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) 触摸屏测试函数，用于测试触摸屏，关键代码如下：

```

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清屏函数，用于清除显示内容，关键代码如下：

```

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画点函数，用于显示触摸坐标点，关键代码如下：

```

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 使用

LCD Touch demo使用步骤如下：

- 1) 编译下载触摸屏案例测试代码。
- 2) 触摸屏进入校准界面，依次点击四个校准坐标点，如下图8。
- 3) 校准完成后LCD会显示校准信息，包含四个校准参数VX、VY、CHX和CHY，如下图9。
- 4) 点击屏幕任意处，将会跳转到触摸屏测试界面，此时触摸屏会将触摸点绘制在LCD上，还会实时显示触摸点的坐标，点击Clear按钮将清除界面，如下图10。

图 9. 触摸屏校准界面



图 10. 触摸屏校准信息

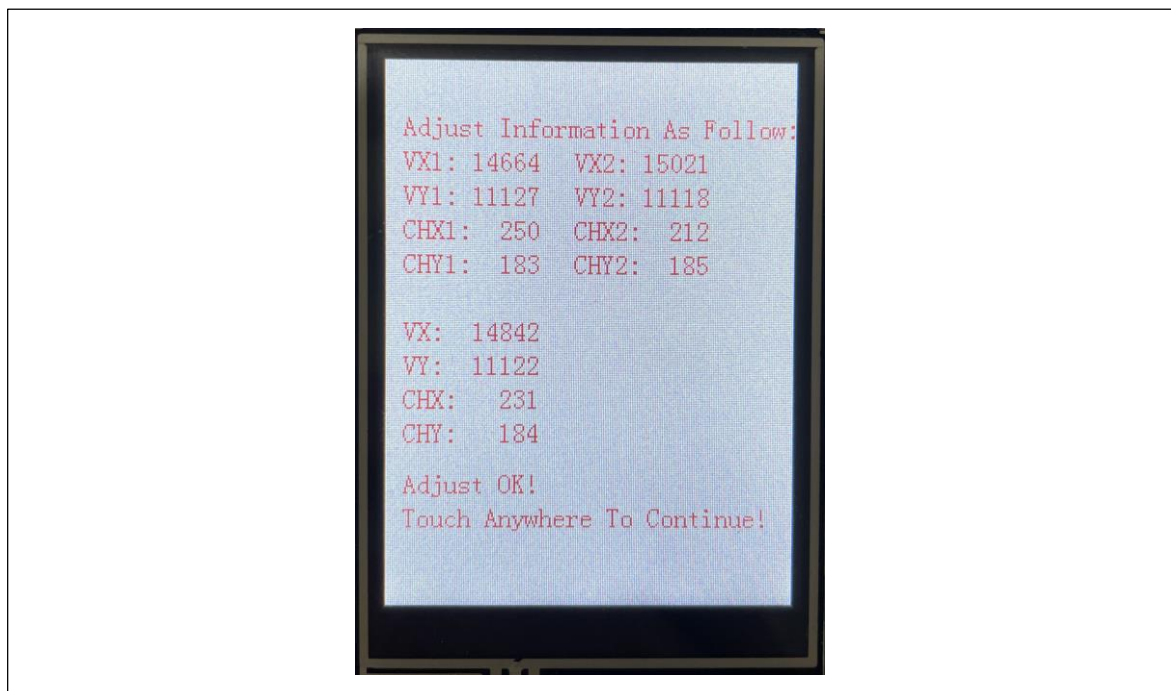


图 11. 触摸屏测试界面



4 版本历史

表 1. 文档版本历史

日期	版本	变更
2022.10.12	2.0.0	最初版本
2022.11.08	2.0.1	增加软件流程章节以及部分关键代码描述

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。

© 2022 雅特力科技 保留所有权利