2025年4月16日 下午 3:24|19分钟 53秒

关键词:

像素、芯片、控制器、电子枪、内存、颜色、引脚、硬件、分辨率、传输、模块、单片机、总线、外接、原理图、传输数据、总线接口、数据引脚

文字记录:

00:09

我们先来讲解不同接口的 LCD 硬件操作原理，在这节视频里，我将会讲解两款LCD，一款是MCU，也就是单片机上所使用的LCD，另一款是MPU，也就是那些能够运行 Linux 系统的比较强大的芯片上所使用的LCD。如果我们只是想去使用LCD，你可以站在应用工程师的角度来看LCD，从这个角度来看 LCD 它是比较简单的。

00:34

首先我们可以认为 LCD 就是由像素点给组成，就比如说在这个图里面它有 y 行像素，每一行里面有 s 个像素点，它的分辨率就是 x 乘以y，这像素点的总个数就是 x 乘以y。我们在屏幕上绘制文字、绘制图片的时候就是去操作这些像素。就比如说你可以在上面。

00:57

来绘制一个字符a，你就把这个 a 所涉及的像素都描成红色，它就可以显示出来了。所以问题归结于我们怎么去操作单个像素。

01:11

首先我们得回答一个问题，单个像素的颜色是怎么表示的？如果我们曾经学过美术的话，应该知道任何一种颜色都可以用红、绿、蓝三种颜料给调出来。在计算机里面它也是这样的，每一个像素的颜色它都可以用红、绿、蓝三个颜色给调出来。在计算机的世界里，一切事物都可以用数据来描述，那么对于某一个像素，它的颜色也应该用数据来描述，那么一个像素，它的颜色用多少位？用多少字节的数据来描述，你可以有很多种选择。这里用一个参数来表示，BPP，bits、per、Pixel，每个像素用多少位来表示？有很多种选择，就比如说我可以用 24 位来表示，也可以用 16 位来表示，甚至你可以用8位来表示等等，我们使用 24 位来表示时，实际上会用到 32 位，其中的8位没有使用。就比如说这种格式，你看最高的8位，它没有使用。

02:08

这个8位用来表示红，这个8位用来表示绿，最低端的这个8位用来表示蓝。你往这 24 位里面写入不同的数值，它就可以表示不同的颜色。我们来举个例子，你可以打开画图板编辑颜色，你看红绿蓝都是 0 的时候，它就是黑色。如果都是最大值， 255 红、255255，红绿蓝三个颜色都是最大值的时候，它就是白。你可以去调整红绿蓝的值，就可以调制出不同的颜色。如果每个像素都用 24 位甚至用 32 位来表示的话，有时候我们会觉得程序会太慢，它太占据资源了。

02:56

我们可以使用 16 位来表示，对于 16 BPP 仍然有两种格式，一种是 RGB 565。从名字上看就可以知道，红绿蓝分别用 5 位、 6 位、 5 位来表示，就比如说这种格式，我们人类对绿色比较敏感，可以分辨出很细微的差别。所以对于绿色，在这种格式里面它使用 6 位来表示，对于红蓝分别使用 5 位来表示，你还可以使用后面这种 RGB 555，这种用的比较少，红绿蓝分别用 5 位来表示，浪费一位。现在我们知道了对于 LCD 上某一个像素它的颜色是怎么表示的，那么我怎么把这些颜色的数据发给这LCD，让它显示出我要求的颜色来呢？不着急，你看来看看这个图。

03:41

LCD 上所有这些像素的数据都保存在一个比较特殊的内存里，这个比较特殊的内存被称为 frame buffer 或被称为显存。已有的芯片手册称为 g RAM， g 就是图形RAM。在显存里面，第0块数据就对应 LCD 上第0个像素，第一块数据就对应第一个像素。如果你想去修改某一个位置的像素，就比如说这个 x y 坐标的这么一个像素，你想去修改它的颜色的话，你需要在显存里面根据它的 SY 坐标找到显存里它的位置，然后把那个颜色写到这个位置上来就可以了。

04:20

有 LCD 控制器会帮你把这个显存的数据搬到 LCD 上去，使用显存的数据来更新 LCD 上各个像素的颜色。作为应用工程师，你只需要去写显存就可以了。剩下的工作都由硬件帮你完成，所以对于应用工程师，对于LCD，我们只需要掌握这三点。第一，在 Framebuffer 里面，每一个像素它用多少位来表示，并且这些数据的格式是怎样的？这第一点，第二这个显存它的基地址是多少？第三我们还需要去掌握一下这个屏幕的分辨率，我根据这分辨率我才。可以在显存里面找到任意像素的位置，知道这 1233 点我们就可以编写出 LCD 的应用程序了。

05:10

那对于我们的驱动工程师我们需要了解的更多，就比如说这个显存，这显存在哪里呀？是在 LCD 里面还是在 LCD 外面？第二之前我们曾经说过有一个 LCD 控制器，这个 LCD 控制器会从 Framebuffer 从这显存里面使用这些数据，用来更新 LCD 上各个像素的颜色，我们驱动工程师很大一部分工作就是去设置初始化这个 LCD 控制器，所以我们需要对这个 LCD 的内部硬件结构了解得更加多一点。

05:46

在这节视频里我试图去讲解不同接口的 LCD 硬件操作原理，我们看看常用的 LCD 接口有哪些？你可以打开 6 U L L 或者 157 的原理图，这 157 的以 157 为例，你看这是它的 LGD 接口，看看里面有哪些接口线，由数据引导，还有什么呢？还有各种控制。这 TP 是触摸屏的，咱不关心。触摸屏的有 v 垂直方向的同步信号， h 水平方向的同步信号，还有pclock，还有 D1 data enable，你看有数据引脚，有这些控制引脚。再来看看对于MCU，对于单片机，对 F103 l c d 控制引角它有哪些呢？来看一眼，我们先把之前那个图给截过来，两个都对比一下。

06:46

左边是157，右边是103，我们来看看对接数据引角，大家都有。你有数据引角，我也有数据引角，好像没什么差别啊。你这边有这些什么水平方向、垂直方向的同步信号，还有pclock，还有 data enable，我这边好像没有了，你看我这边有什么呢？有片选信号、有写信号、有读信号、有背光信号，还有一个 RS 信号，干嘛用的？咱们也不知道。

07:21

你看这两款LCD，它们的信号基本上不一样，除了大家都有 data 引脚之外，其他引脚大家都不一样，那怎么去理解这些不同接口的 LCD 呢？虽然这些 LCD 的接口不一样，但是它们的内部实质是一样的。我针对这不同接口的 LCD 抽象出了一个统一的硬件模型，怎么去理解这个硬件模型呢？我们来看一看。好，我先把这个硬件模型复制一下。

07:59

先来看看第一种比较简单的单面积上所使用的。对于单面机这 frame buffer 显存，还有 LCD 控制器，还有LCD，它们都放在一块组装成一个 LCD 模组。 LCM 我们的单面积就比如说 103 F， 103 就直接跟他们来通信。这第一种情况，第二种情况是下面这种，你看 LCD 控制器还有内存跟这个 LCD 是分隔开的， LCD 就只是 LCD 控制器，是位于 arm 芯片内部的，这 arm 芯片它可能是外接内存，在这个外接的内存里面分配出一块空间，这就是显存，就是 frame buffer。这是第二种情况。第一种情况就对应我们的 103 单板机，第二种情况就对应157，对应 imx6ull 这些能够运行 Linux 的比较高性能的芯片。

09:14

这种接口一般来说称为 8080 接口，这种接口被称为 TFT RGB 接口，这两种接口当然是不一样的。但是你看 8080 接口进入这一边这个模块， LCN 模块和下面这整个模块，它们的原理是完全一样的。我们先来看看比较简单的这个 8080 接口 F103 或者称为 MCU 单片机的性能比较弱，所以它外接这些模块用什么接口最好呀？用内存的接口最好，我们去访问内存时，一般来说用什么数据线，假设这边是MCU，也就我们的。 SM7 这边是内存SRAM，它们之间的接口有哪些？ MCU 要去读内存？我怎么知道你要读有个读信号。 MCU 要写内存，我怎么知道你要写呀？有个写信号，你要读写我哪个地址上的数据，所以这里面会有地址线 address 总线有不止一条地址线，这数据怎么传输呀？怎么从 MCU 传到SLM？或者怎么从 SLM 传到MCU？有个 data bus， data bus 有可能不止一条的线，8位的话有 8 条线， 16 位的话有 16 条线， 32 位的话有 32 条线。

10:55

好，我这 MPU 可能接有SRAM，也有可能接有第二块SRAM，我怎么去分辨这不同的SRAM？这还有片选信号 CS 0、CS1，所以你看对于 MCU 访问内存这样的接口，它至少得有这么一些引脚。好，那回到我们的LTE，难道你想让我这个 LCM 这模组也有那么多线吗？数据线。好，我不得不保留，我要传输数据嘛？那地址线你也要我保留那么多吗？我这个接口越多，越多越多，我设计起来越麻烦，买我的人就越少。所以我能不能够把这个地址线精简一下？可以的。

11:57

你看对于 LCM 的话，它有什么线片选肯定是必须有的， CS 读信号肯定有的，写信号肯定也是有的，还有 data bus 也是有的。好，你可以把数据发给我。那么你怎么发地址给我？你要访问我这个显存里面哪一个地址呀？我怎么把地址发给你呀？我要把地址作为数据发给你。

12:30

那问题就来了，在 databus 上到底传输的是数据本身，还是其他的地址，还是其他的命令？我需要有一个引角来分辨，这引角就是 data o command，影响这个 data command 引角等于 0 或者等于一的时候，分别来表示上面传输的是数据本身还是其他信息。你看使用这样的接口我们就可以省去一些 address bus，让这个引脚稍微少一点，对于硬件设计来说就会简单很多。

13:07

对于 8080 接口的 LCM 模块，看这 LCD 控制器和显存，它们是绑定在一块的，它是由一个芯片集成了显存，集成了 LCD 控制器 103 或者称为MCU，它要跟这个芯片打交道，它可以通过 data bus，然后配合这个 data command 引脚把命令把地址发给这个芯片，然后再把数据发到这个显存上面去，由这个芯片内部的 LCD 控制器把这个显存的数据用来刷新 LCD 屏幕。对于这种情况我们不需要去了解这个芯片的内部结构，我们只需要去根据这个芯片的接口发出特定的命令，发出地址、发出数据就可以了。

13:56

对于这种接口的 LCD 模块，它使用上很简单，但是它有个缺点，就是这个 LCD 的分辨率没办法做得很大，一旦做得很大的话，这个显存就要很大。这写着很贵的，所以使用 8080 接口的这 LCD 分辨率一般来说都不会太大。这种模组它的内存使用的一般是SRAM，它比较贵，那么我想支持比较大的分辨率，那怎么办呢？你可以使用下面这种方案，它的内存可以使用 DDR 或者 SD RAM 等等稍微便宜点的内存，在这些内存里面我可以分配出很大的显存。对于这种方案它需要有一个强大的芯片，这个强大的芯片要能够支持DDR，能够支持HDR，它甚至把 LCD 控制器也给集成进来了，这就是MPU。一般来说能够跑 Linux 的都是使用这种方案。你看在 arm 芯片内部，它集成了 LCD 控制器，在 LCD 控。控制器，它可以去外部的内存里面得到这个 frame buffer，得到显存里面的数据，它会自动地从这显存里面得到数据，用这些数据来更新 LCD 屏幕，它跟 LCD 的接口就是这些接口线，怎么去理解这接口线呢？要想去理解这主线，我们需要想象一下，你认为在屏幕后面有一个电子枪，这个电子枪可以把颜色打到某一个像素上面去，那问题就来了，我这个电子枪只有一把，我电子枪把颜色打到这个像素去之后，它怎么才可以移动到下一个像素？移动一个像素由 dclock 来控制，每来一个 dclock data clock，这个电子枪就移动一个像素。

15:58

然后第二，我这个电子枪移动到最右边的时候，它应该跳回到最左边，它怎么跳呀？从最右边跳回来，跳到下一行，谁来控制我这个电子枪跳到下一行的行首呀？由 ASYNC 水平方向的同步信号每来个脉冲，我这个电子枪就跳到下一行的行首。好，第三，我这个电子枪已经把数据打到最后一个像素了，它应该跳到最前面，跳回一帧的开始这里它怎么跳呀？VSSYNC，每来个VSSYNC，这电子枪就从这里跳到最上面一行，好，再来看看。第四，你说这个电子枪会把数据打到这个像素上面去，那数据从哪里来呀？数据来自哪里？来自 RGB 三主线？来看看我们之前曾经看过这个原理图，对157，你看由 data 0 到 data 23 这 24 条线就分别对应 RGB 三主线。好，还有一条隐角，第一 data enable 在这个电子枪从最右边跳回到最左边，或者从最下边跳到最上边的时候，这个数据应该是无效的，由 data enable 来决定来控制电子枪是否获得数据，是否发出颜色。你看我们使用电子枪的概念就非常容易理解这些信号线。

17:56

好了，我们现在已经理解了 8080 接口或者 RGB 接口的这些信号为什么是这么组成的，那以后如果我们想去编写 8080 这 LCD 模块的驱动程序时，重点是去了解 LCD 控制芯片，你需要去阅读这个芯片手册，来看看怎么给它发命令，怎么给它发地址，然后怎么把数据发到这个显存里面来。你还可以发出某些命令来控制这芯片，使能这 LCD 来设置它的背光等等等等。

18:27

那么对于 RGB 接口，你也需要去阅读这芯片手册来设置这个 LCD 控制器，就比如说它可以发出DCLOCK，这 DCLOCK 的频率有多少？你需要根据外接的 LCD 来确定这个 dclock 的频率，根据这个频率反过来去设置这 LCD 控制器。最重要的是你需要去分配 frame buffer，把这 frame buffer 的首地址告诉这 LCD 控制器。

18:52

好，这节视频我们简单的讲解了这两种接口的LCD，讲解了它们的硬件原理，实际上这两种接口的实质是一样的，它们都可以归入一个标准，这个标准被称为 MIPI 标准。你看对， 8080 接口，那就是 MIPI DBI 接口，显示总线接口。为什么叫总线呢？你看有databus，有 data command 引脚来分辨上面传输的是数据本身还是其他命令，其他地址叫总线接口，还有一种 display Pixel INTERFACE。 Pixel 我直接传输的是像素里面的数据直接控制的是像素。对于这种接口，它上面传输的就是比较原始的像素信号。还有一种接口我们还没有涉及DSI，它可以使用更少的数据线来传输数据，这种在手机里面会用得比较多，现在我们还没有涉及。好，这节视频我们就讲到这里。