第六章 CPU 你省省心吧

“CPU 运行时间是宝贵的资源，我们要把有限的 CPU 时间投入到更有意义的事情中去。”

在我们进行嵌入式开发的过程中，你一定干过这几件事：用 GPIO 模拟某种通信接口，比如 SPI 等；用空循环来实现延时 delay；空等寄存器的关键状态位。也许是出于无奈，比如所使用的芯片没有硬件 SPI 或通道不够，亦或者此时 CPU 除了空转并没有其它事情要作，但是我们一定要有这样的意识：这是在浪费 CPU 资源。

CPU 是嵌入式系统的核心，但是它不必深入参与到每一个细节中去，记住：CPU 是片上所有硬件资源的统领者，而非事必躬亲的苦力。我们要学会尽最大可能充分利用片上硬件资源，甚至在芯片外部扩展一些专门的硬件电路来完成功能设计。本章振南将通过几个实例来向大家说明如何减轻 CPU 负担，而用片内片外的硬件来实现我们想要的功能。

1、石油测井仪器

1.1 背景知识

在我的职业生涯中，有 5 年多的时间在作石油仪器。这是一个很传统的行业，但也是非常综合性和吃技术的行业。

有人说：“你这一章似乎要讲的是 CPU 的利用率问题，怎么又讲起石油仪器来了？”别急，振南自我用意。

请看图 6.1。

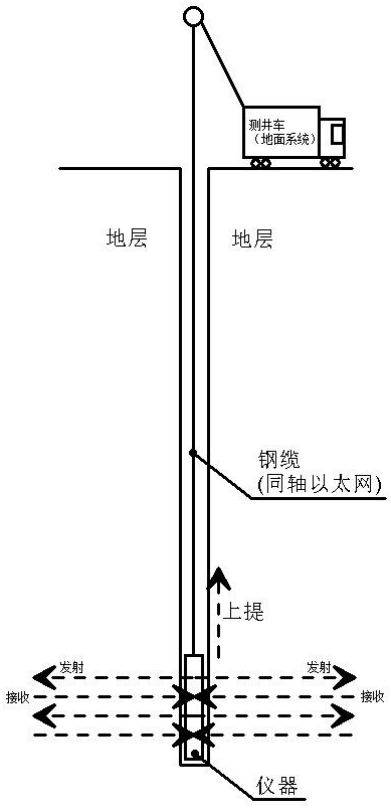

图 6.1 石油测井系统示意图

上图所示为石油测井系统的简易拓扑示意图。工作时测井车通过轮盘拉动钢缆上提，与此同时仪器向外发射信号(电或超声)，并接收返回信号经过计算将结果通过同轴以太网上传到地面系统，由上位机绘制出曲线。最终曲线将交给解释工程师，来判断储层的位置。

上提的速度是一定的，我们当然希望在某一个深度上多采一些数据，即尽量提高采样率。这样最终的测井曲线上就能体现出更多的细节。

OK，这就是最基本的原理和背景。

1.2 测井数据采传的实现

电路上比较清晰，如图 6.2。

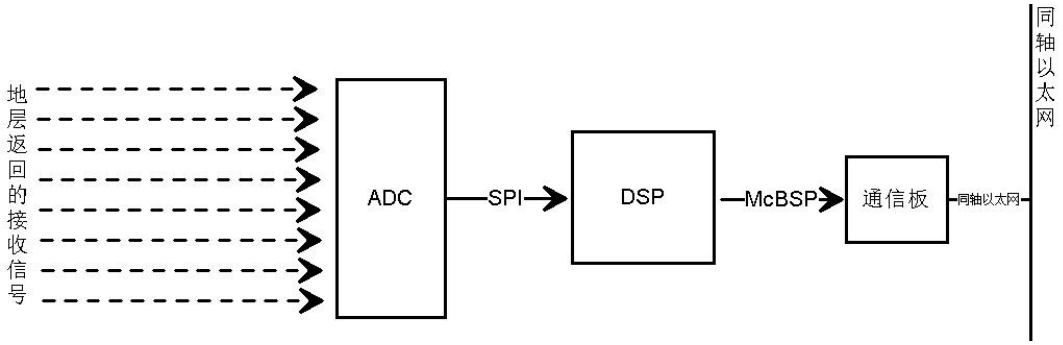

图 6.2 测井仪器数据采传原理框图

1.2.1 最直接的方案

最直接的方案是所有人都能想到的方案，就是采集、计算、发送按部就班的进行，如图 6.3 所示。

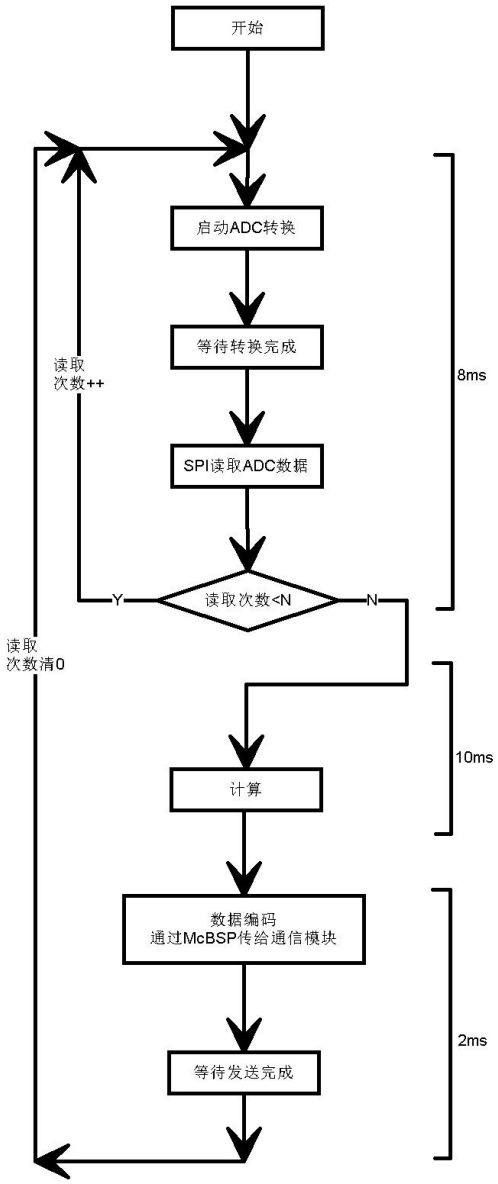

图 6.3 测井数据采传最直接的实现方案

每一个周期要作的事情就是：ADC 采集一段波形，然后进行计算，主要是一些数字滤波、FFT、DPSD 之类的 DSP 处理，最终将结果数据按协议格式打包通过 McBSP(TI DSP 专有的通信接口)发送给同轴以太网通信模块。我们当然希望这个周期越短越好，这需要将一些步骤优化压缩。

1.2.2 加入 DMA 的优化方案

上面的方案，仔细看一下就会发现，它的所有操作都是需要 CPU 参与的，大量的时间都在等待外设。如何降低 CPU 的参与度，把其宝贵的时间不要浪费在空等上，而放在核心算法的计算上，请看图 6.4 所示。

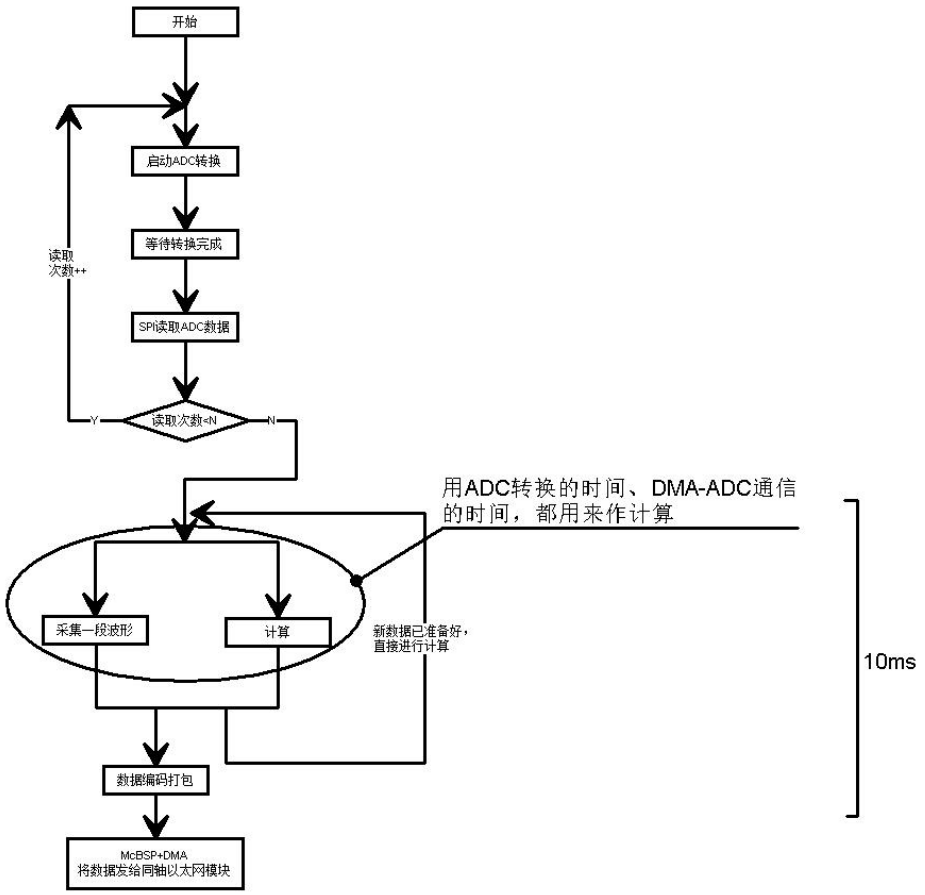

图 6.4 加入 DMA 的数据采传优化方案

我们首先由 CPU 参与完成一次波形采集，然后开始针对采集数据进行计算，因为涉及大量浮点数据的数字信号处理，所以计算过程会比较花时间，一次计算大约需要花费 10ms。与此同时，我们适时的不断启动 ADC 转换，在其转换的时间间隙里进行计算，然后直接启动 SPI-DMA 传输来读取 ADC 的转换数据，而 CPU 不用去等 DMA 传输完成，可以利用 DMA 传输的时间进行计算，最后回过头来立即进行下一次计算，因为此时新的波形已经准备好了。这样，一个周期的时间可以压缩到 10ms，采样率比原来提高了一倍。

振南是想通过这个实例来告诉大家：CPU 的运行时间是宝贵的，将片上的硬件资源充分的用起来将可以释放出更多的 CPU 时间来作更有意义的事情。一些技巧和 DMA 的合理运用是行之有效的办法。

其实很多时候能被用来发挥的硬件资源并不只限于片内，我们自己设计一些简单的片外电路加以辅助，有时候可以达到意想不到的效果，请往下看。

2、巧驱摄像头

2.1 摄像头时序分析

我知道很多人都对摄像头模块感兴趣，想用单片机驱动一下试试效果，但是作成功的并不多，如图 6.5。

图 6.5 比较盛行的 OV7670 摄像头模块和模组

究其原因有几点：1、摄像头 CMOS 芯片的时序较为复杂；2、SCCB 通信及相关寄存器的配置；3、时序过快，而且是按其固有频率主动输出，难于捕捉和采集数据。

它的时序有多快，我们来看下图，如图 6.6 所示。

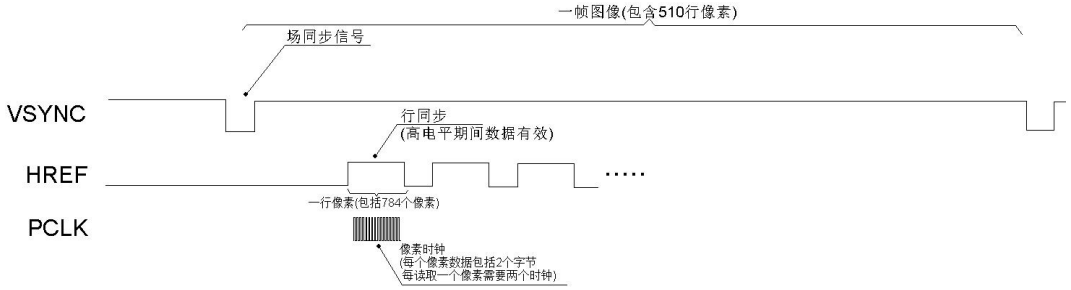

图 6.6 OV7670 的时序示意图

OV7670 在 VGA 模式可达到的最高帧率为 30fps，即每秒钟产生 30 帧 640X480 尺寸的图像。从官方资料上得知 VGA 模式下实际输出的行数为 510，每行输出的像素数为 784(多出来的行数与像素数是多余的，其数据是无效的，我们只关注 HREF 为高电平期间的像素数据)。这样，PCLK 的时钟周期为 1/(30*510*784*2)=41.7ns。想要用一般单片机的 GPIO 来直接采集像素数据，几乎是不可能的，因为 IO 与 CPU 的速度都不够快。

2.2 使用 DCMI+DMA

要读取摄像头如何高速的数据，必须要有专门的硬件。我们可以选用 ST 的 STM32F4 系列 MCU，它内置了 DCMI(数字摄像头模块接口)，使用它将可以很轻松的完成图像获取的功能。它要配合 DMA 来工作，如图 6.7。

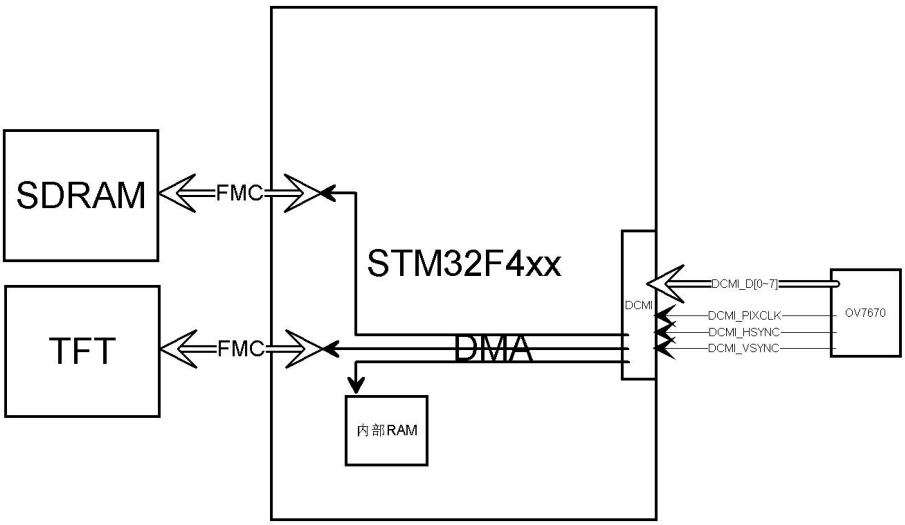

图 6.7 使用 DCMI+DMA 实现对摄像头的驱动

DCMI 获取摄像头数据，可以通过 DMA 直接将数据保存到内部 RAM 或外部的 SDRAM，甚至直接写入到 TFT 中，实现图像的实时动态显示。而在整个过程中，CPU 只不过在作一些配置性的工作，并没有参与图像数据采集和传输。所以，用高端芯片会使我们的开发工作如虎添翼，事半功倍，就是因为它有更强大的硬件外设来为我们完成特定的功能实现。当然，更强大的硬件也意味着更多的学习成本，我们需要仔细学习如何正确的使用它来达到想要的效果。

有些时候，硬件外设电路甚至比 CPU 内核更复杂，比如有些多媒体编解码 SOC，CPU 内核只是 51 或 M0，片上更大的面积是诸如 H263 之类的编解码电路。所以，作嵌入式开发的工程师，首先要充分了解自己手上有哪些硬件资源，而不要所有功能都纯依靠 CPU 来实现。

2.3 自搭外部电路

本节的名字是“巧驱摄像头”，上面所介绍的方案都不算不上一个“巧”字。上述方案中必须要求单片机有 DCMI 之类的专用硬件，那不用 DCMI 可不可以？比如拿普通的 51 或低端的 M0 单片机，可不可以实现对摄像头的驱动。答案是肯定的，不过这需要我们在外部电路上作些手脚，如图 6.8 所示。

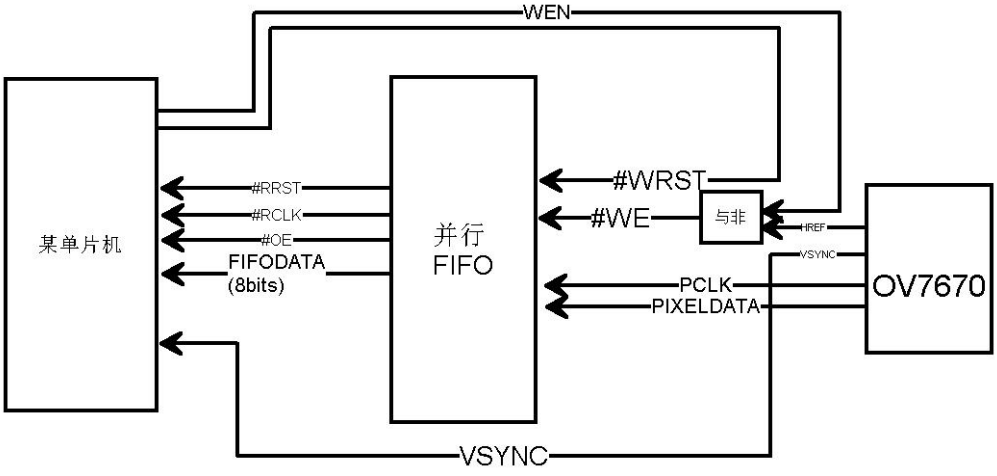

图 6.8 通过片外并行 FIFO+时序调理实现图像采集

配合下面的流程图，大家就知道其巧妙之处了，如图 6.9。

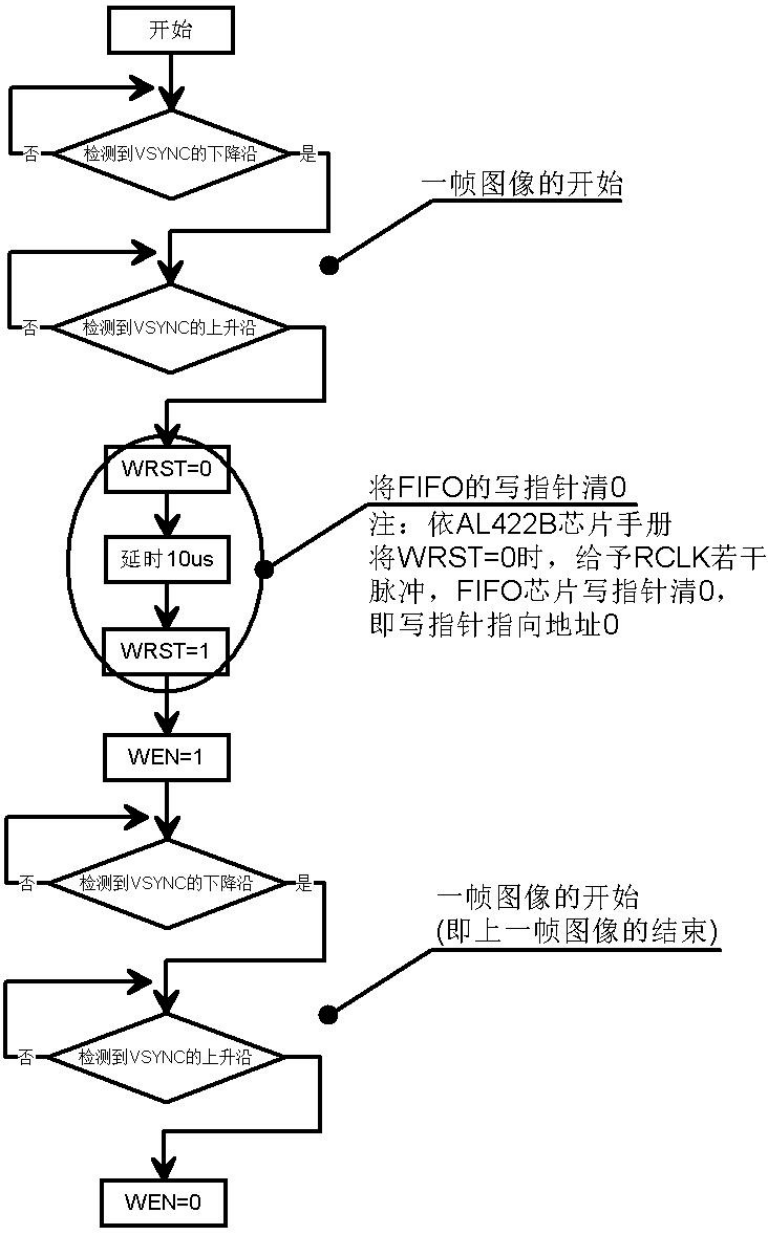

图 6.9 通过片外并行 FIFO 实现一帧数据的获取

程序按上图描述的逻辑运行之后，一帧图像就存到 FIFO 中了。此时单片机可以慢慢从读取端(并行 FIFO 分为写入端与读取端，分别对应的有写指针与读指针)读到图像数据了。这样 CPU 和 IO 的速度就再也不是瓶颈了。通过这样的机制，任何单片机都可以轻松实现图像采集了。

在此过程中，CPU 都干了什么？似乎只有等待帧同步信号 VSYNC 和操作几个 IO。这种方式比 DCMI+DMA 更省 CPU(DMA 实际上会占用一半的片内数据总线带宽，使 CPU 的运行效率降低)，而且更灵活，对单片机硬件的依赖更小。

3、巧驱 7 寸液晶屏

通过上面几个实例，大家应该知道振南所谓“巧驱”的路数了吧，对，就是多让硬件说

话，我们要作“软硬兼施”的工程师。

OK，如果我问大家：“我能用 51 或 M0 单片机，驱动 7 寸大屏液晶(800*480)，如图 6.10，并且流畅播放视频，你信不信？”你一定会说：“不太可能吧，屏率速率不够。”但我既然这么问，那振南一定是已经实现了，这里我就把实现过程给大家讲一下。

图 6.10 7 寸 TFT 液晶模块

先来看原理图，如图 6.11~6.14。

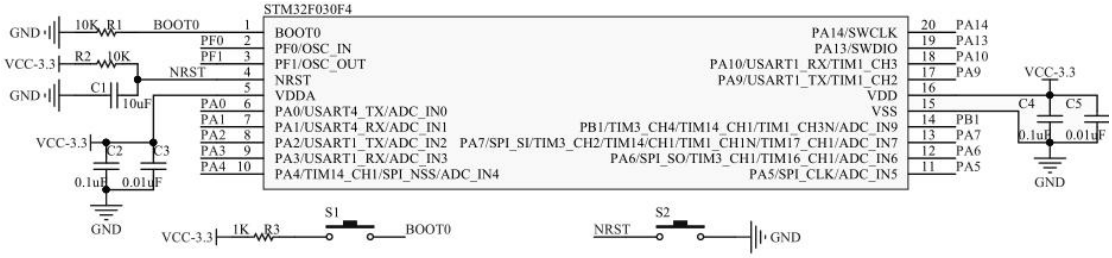

图 6.11 巧驱 7 寸液晶屏原理图之 MCU 部分

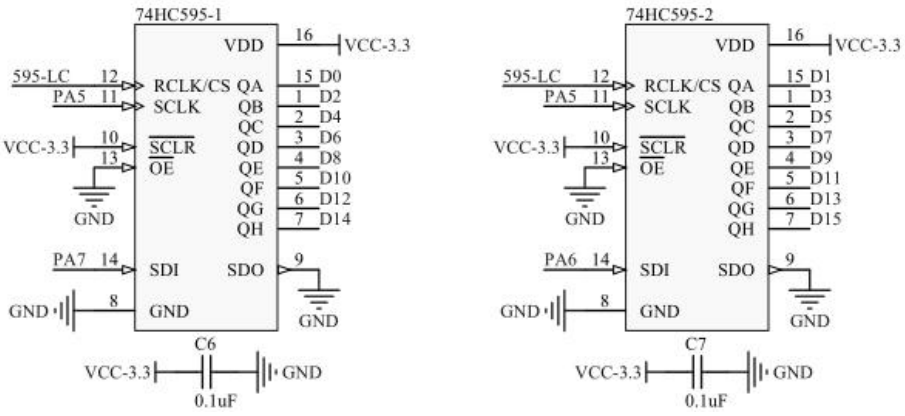

图 6.12 巧驱 7 寸液晶屏原理图之 74HC595 串转并部分

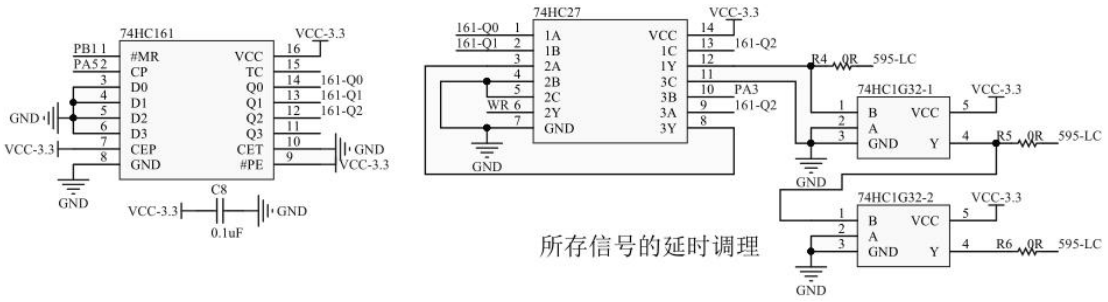

图 6.13 巧驱 7 寸液晶屏原理图之八 8 进制计数与时序调理部分

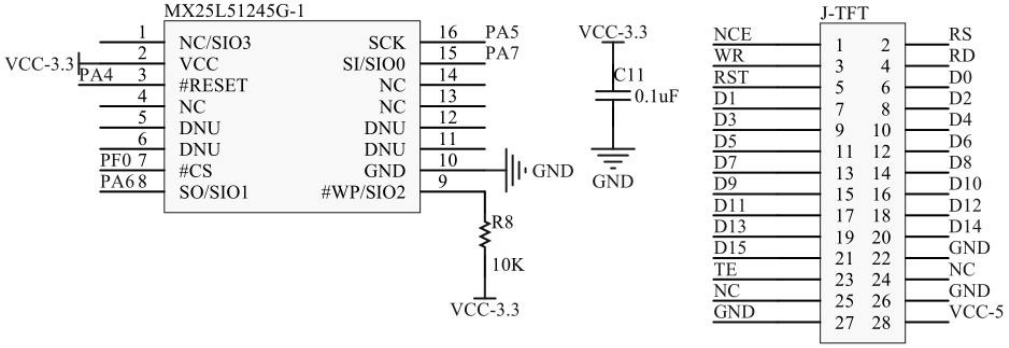

图 6.14 巧驱 7 寸液晶屏原理图之 spiFlash 与 7 寸 TFT 接口部分

基本的实现逻辑如图 6.15 所示。

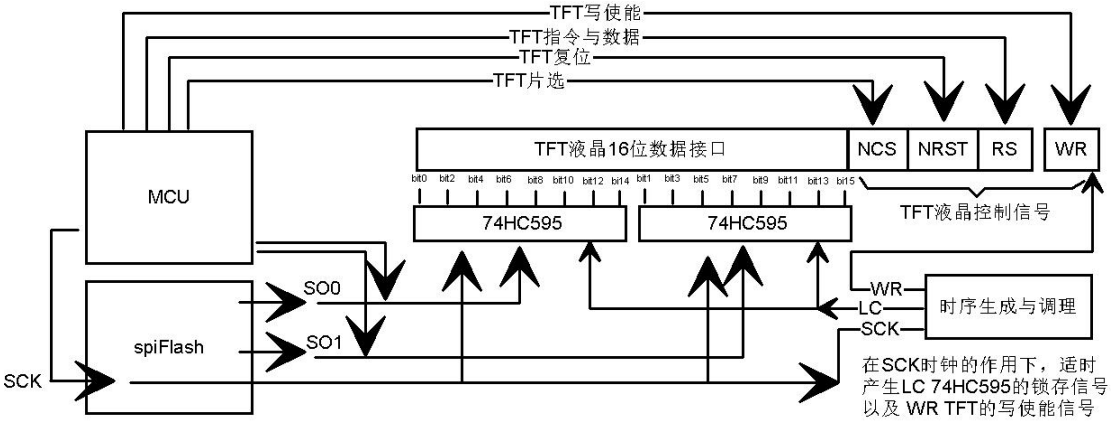

图 6.15 巧驱 7 寸液晶屏之基本实现逻辑框图

仔细观察上面的原理图与逻辑框图，估计很多人已经明白了振南的意思，振南再给出配套的流程图，逻辑就更清晰了，如图 6.16 所示。

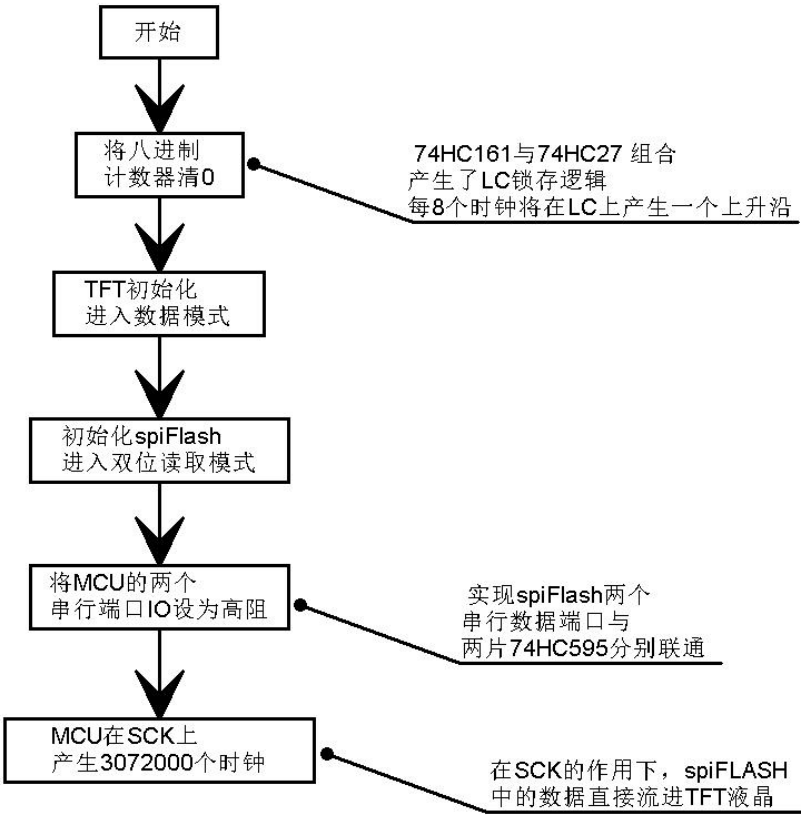

图 6.16 巧驱 7 寸液晶屏的基本流程图

两片 74HC595 用于将 16 位串行数据转换为并行，与 TFT 液晶的 16 位数据接口相连。74HC595 的串行数据输入同时与 MCU 的两个 GPIO 以及 spiFlash 的两个串行数据端口相连。当 spiFlash 失能时(即 CS 置高)，其数据端口呈现高阻，此时 74HC595 可由 MCU 操作；而当 MCU 的 GPIO 设置为高阻时，两片 74HC595 可分别接收来自 spiFlash 的双位串行数据。这样的复用设计，可以使 MCU 对 TFT 液晶进行预先的初始化，使其工作在纯像素数据写入的模式；而在高速数据写入的阶段，MCU 退出而让 TFT 接收来自 spiFlash 的数据。

两片 74HC595 实现串转并的要点在于 LC 锁存信号的产生，每产生 8 个 SCK 脉冲，则自动产生一个 LC 上上升沿，这是时序生成与理调逻辑的一部分。实现的根本在于 74HC161 与 74HC27 的组合运用，如图 6.13。首先对 74HC161 复位清零，此时[Q2:Q0]=000，74HC27 是三输入或非门，其输出 1Y，即 595-LC 为 1；时钟的输入后[Q2:Q0]随之自增 001、010 …… 在 000 之前 595-LC 均为 0，而 8 个时钟之后，595-LC 将变为 1，即产生了上升沿。这里振南给 595-LC 增加了两级 74HC1G32 作为缓冲，为的是增加一些延时，以使 74HC595 的存锁数据输出更稳定。

然后是液晶的 WR 信号的产生：从图 6.12 中可以看到，WR 信号是一个 GPIO 与八位计数器输出最高位 Q2 的或非非(没错，是或非非)。当 Q2 为 0 时，WR 受控于 GPIO，此时可用于 MCU 对 TFT 预先进行初始化操作。当 GPIO 为 0 时，WR 受控于 Q2，每 8 个时钟会产生一个下降沿(前面那个或非非是为了推迟一下这个下降沿，以使 16 位并行数据写入液晶更稳定)，并维持 4 个时钟周期。

基本的要点已经描述清楚了。至于时钟的产生，唯一的要求是要产生特定数量的时钟，而不能是连续不断的。比如一帧图像的数据量为 800*480 半字，我们要输出 3072000 个时钟才能让一帧图像显示到液晶上。所以我们不能用 MCO 或者是 PWM，而要用 SPI，如果是 8 位 SPI，要写 384000 次，如果是 16 位 SPI，则要写 192000 次。当然，为了节省更多的 CPU

资源，我们可以使用 DMA。当时钟不断的产生，一帧帧的图像显示到液晶上时，视频就流畅的播放出来了。

好了，本章用 3 个实例阐述了本章最开头的那句话：CPU 时间是宝贵的，我们要把有限的 CPU 时间投入到更有意义的事情中去。实际开发中，充分地利用硬件资源，自行灵活扩展一些硬件电路，通常可以达到意想不到的效果，甚至可以化不可能为可能。

永远记住：我们很多时候作的是嵌入式软件的工作，但归根结底我们搞的是硬件。