第五期：

1、gc2053 sensor datasheet i2c时序图讲解

2、mipi协议的cci之i2c读写操作时序

3、gc2053 sensor driver代码电源管理注册和sensor源码gc2053\_probe接口讲解

-> v4l2\_async\_register\_subdev\_sensor\_common

-> v4l2\_async\_subdev\_notifier\_register

-> v4l2\_async\_register\_subdev

Linux内核从3.x版本之后开始支持使用设备树，这样做的意义重大，可以实现驱动代码与设备的硬件信息相互的隔离，

减少了代码中的耦合性，在此之前，一些与硬件设备相关的具体信息都要写在驱动代码中，如果外设发生相应的变化，

那么驱动代码就需要改动。但是在引入了设备树之后，这种尴尬的情况找到了解决的办法，通过设备树对硬件信息的抽象，

驱动代码只要负责处理逻辑，而关于设备的具体信息存放到设备树文件中，这样，如果只是硬件接口信息的变化而没有驱动逻辑的变化，

开发者只需要修改设备树文件信息，不需要改写驱动代码。

Runtime PM (Runtime Power Management)翻译过来就是运行时电源管理。主要的作用是:

每个设备处理好自己的电源管理，在不需要工作时进入低功耗状态。也就是"各人自扫门前雪"。

of\_match\_ptr

SET\_RUNTIME\_PM\_OPS：

对于runtime PM，默认状态下设备的状态是suspend，如果硬件上它是运行状态，

需要调用pm\_runtime\_set\_active()来修改它的状态，然后调用pm\_runtime\_enable()来使能runtime PM。

一般是在probe()的结尾处使用，以为它可能导致runtime的suspend/resume函数立即调用。

一般在驱动remove中调用pm\_runtime\_disable()

to\_i2c\_client：

to\_i2c\_client函数的作用是根据给定的设备结构体指针，返回对应的I2C设备客户端结构体指针。

这个函数在内核驱动程序中经常被用来获取I2C设备的相关信息并进行操作。

具体来说，to\_i2c\_client函数接受一个指向设备结构体的指针作为参数，然后通过内部的转换逻辑，

将其转换为I2C设备客户端结构体指针。

这样，驱动程序就可以使用返回的I2C设备客户端结构体指针来访问和操作I2C设备的属性和功能。

i2c\_get\_clientdata：

i2c\_get\_clientdata函数的作用是根据给定的I2C设备客户端结构体指针，返回该结构体中存储的私有数据指针。

这个函数在内核驱动程序中经常被用来获取I2C设备客户端结构体中存储的私有数据，以便在驱动程序中进行操作。

具体来说，i2c\_get\_clientdata函数接受一个指向I2C设备客户端结构体的指针作为参数，然后通过内部的转换逻辑，

将其转换为存储在该结构体中的私有数据指针。

这样，驱动程序就可以使用返回的私有数据指针来访问和操作I2C设备客户端结构体中存储的私有数据。

to\_gc2053：

to\_gc2053函数的作用是根据给定的V4L2子设备结构体指针，返回对应的GC2053图像传感器设备结构体指针。通过这个函数，

驱动程序可以使用返回的GC2053图像传感器设备结构体指针来访问和操作GC2053图像传感器的属性和功能。

of\_match\_ptr宏定义的作用是将一个设备节点的驱动程序数据结构指针转换为一个of\_device\_id结构体指针，

以便进行设备匹配。of\_device\_id结构体定义了设备树中设备节点与驱动程序之间的匹配规则。

KERNEL\_VERSION是一个宏定义，用于表示内核版本号

内核版本号通常由三个数字组成，分别表示主版本号、次版本号和修订版本号。

例如，内核版本号3.18.1中，3表示主版本号，18表示次版本号，1表示修订版本号

主版本号：0x000000 << 16 = 0x00000000

次版本号：0x000001 << 8 = 0x00000100

修订版本号：0x000001

DRIVER\_VERSION = 0x00000000 | 0x00000100 | 0x000001 = 0x00000101

dev\_info函数来输出驱动程序的版本信息。dev\_info函数的第一个参数是指向设备结构体的指针，

表示要输出信息的设备。第二个参数是一个格式化字符串，用于指定输出信息的格式。

gc2053的默认状态从左下角读出，引脚A1位于左上角。

由于图像被镜头垂直和水平倒转，因此当引脚A1位于左上角时，

图像输出正常；读出方向可以由寄存器设置

像素阵列被bayer图案滤色器覆盖。原色GR/BG阵列采用线交替排列方式

如果列中没有翻转，则从0到1919读出列。如果在列中翻转，列被读出从1919到0

如果row中没有filp，则从0到1079读出row。如果翻转为row，则从1079读出row到0

透镜主射线角

彩色滤光片的光谱特性

彩色滤光片的光谱如下所示

6.1.2：cci(i2c)读写操作：

在从随机位置读取的单次操作中(参见图4)，主机对所需的INDEX执行一个虚拟写操作，发出一个repeat START条件，

然后用读取操作再次寻址从机。

在确认其从机地址之后，从机开始向SDA线路输出数据。主机通过设置否定确认和STOP或repeat START条件来终止读操作。

通过对从机进行读操作寻址，也可以从最后使用的INDEX读取数据(参见图5)。

从机通过将数据从最后使用的INDEX发送到SDA线来进行响应。

主机通过设置否定确认和STOP或repeat START条件来终止读操作。

从随机位置开始的顺序读取如图6所示。主机对所需的INDEX执行一个虚拟写操作，

在从机得到确认后发出一个repeat START条件，然后用一个读操作再次对从机进行寻址。

如果主机在接收到数据后发出确认，这将作为从机的一个信号，

表明读取操作将从下一个INDEX继续。当主机读取最后一个数据字节时，它发出一个否定的确认和一个STOP或repeat START条件。

从当前位置开始的顺序读取(参见图7)类似于从随机位置开始的顺序读取。

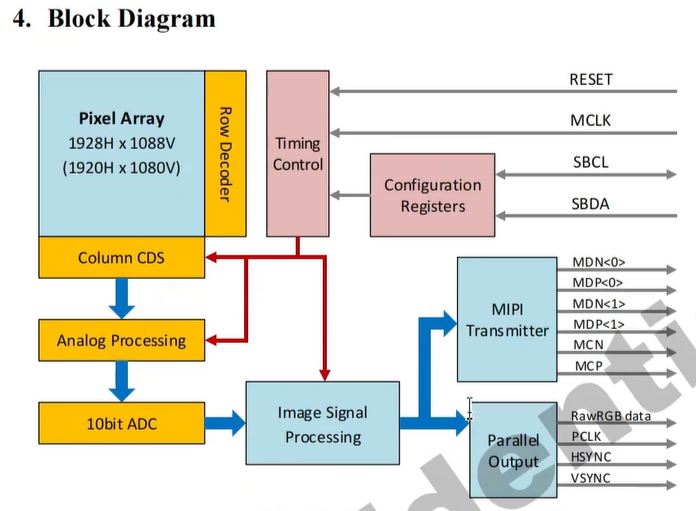
唯一的例外是没有虚拟写操作。主机通过发出一个否定的确认和一个STOP或repeat START条件来终止读操作。

对随机位置的写操作如图8所示。主机向从机发出写操作，

然后在从机确认写操作后发出INDEX和数据。写操作在主机发出停止或重复启动条件后终止

从随机位置开始的顺序写入操作如图9所示。从机在接收到每个数据字节后自动增加INDEX。

从随机位置开始的顺序写操作将在主机的STOP或repeat START条件下终止。



通过I2C控制Configuration Registers(控制寄存器)，然后控制时钟等配置。

Pixel Array是像素矩阵。这个是1080p的。

Row Decoder:行的解码器，是硬件上的电路。

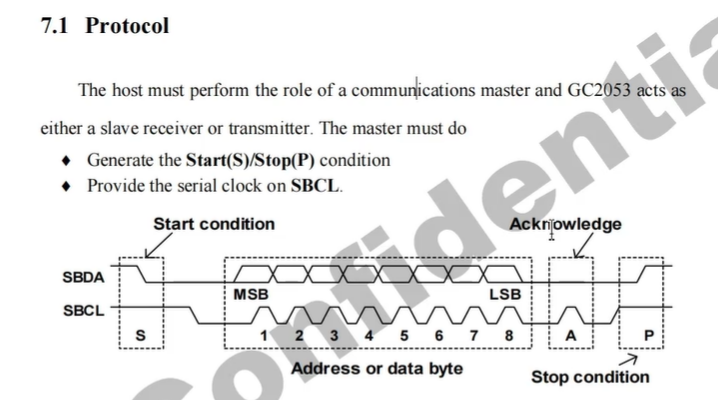
Column CDS：是采样的一个电路。

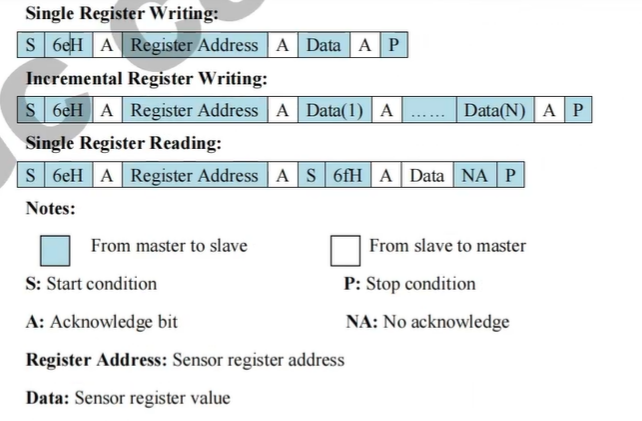
也就是说最原始的sensor经过一个采样，就会转化为原始的模拟信号（Analog processing）,最原始的模拟信号再通过这个10位的ADC模数转换，转换为数字信号。

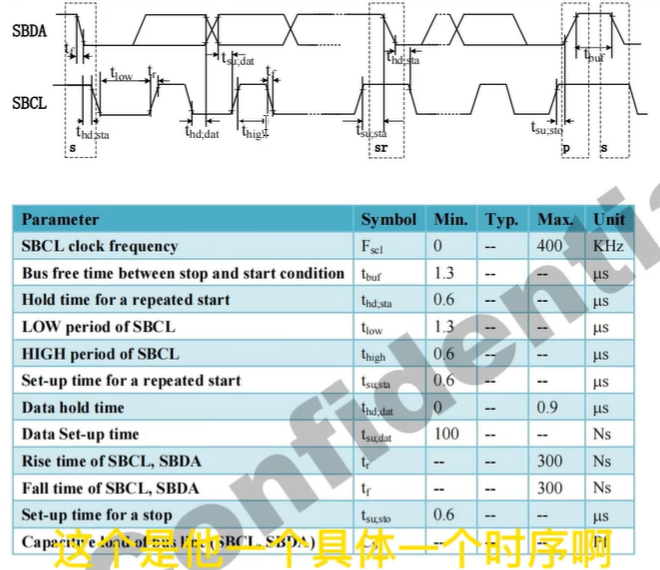
数字信号经过Image Signal Processing的一个处理，其实是经过ISP的一个处理。经过处理的数据可以通过两种方式输出。

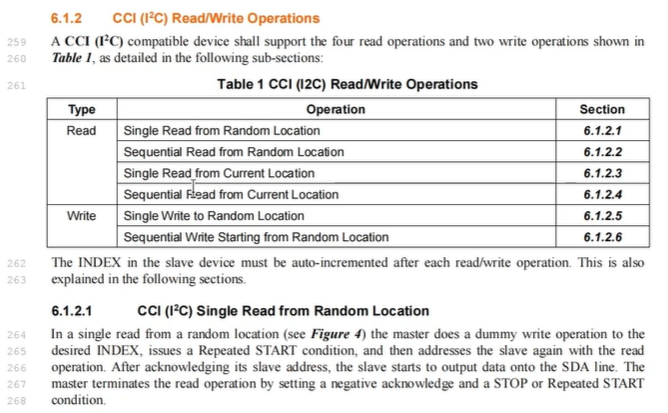
Parallel Output是直接的平行输出。

MIPI Transmitter是通过MIPI的输出。后面的那些是输出的管脚配置。









"C:\Users\zhongqing\Desktop\AI人工智能学习笔记\嵌入式音视频\MIPI标准文档大全.rar\_mipi协议资源\MIPI标准文档大全\mipi\_CSI-2\_specification\_v2-1-2018\mipi\_CSI-2\_specification\_v2-1-2018.pdf"

