对于没有调试过 gsensor 的人来说，首先要了解一下设备树，也就是我们常说的 dts 或者 dtsi 文件，此文件主要是包含了一些设备信息，也可以根据自己所调试的模块，添加对应的设备。MTK 平台比较早的版本（Android 大版本），里面还没有引入，通过设备树注册相关的设备，驱动加载前，会通过标准接口先注册设备，之后通过设备的 name 匹配驱动。设备树的引入，则会在其展开的时候，就加载了所有的设备，对应的驱动也是通过 name 匹配，最后执行 probe 函数，完成驱动的注册。

1. 设备树简要介绍：

平台的 msm8909.dtsi 文件中，包含 5 组 i2c 资源，分别命名为 i2c_n(n=1,2,3,4,5)，而且对于每一组 i2c 资源都有配置 OK，我们调试的 I2C 设备，只要在对应的 I2C 总线下，添加上调试模块的设备信息就可以了。具体有关内容见下图：

```
aliases {
  /* smdtty devices */
  smd1 = &smdtty_apps_fm;
  smd2 = &smdtty_apps_riva_bt_acl;
  smd3 = &smdtty_apps_riva_bt_cmd;
  smd5 = &smdtty_apps_riva_ant_cmd;
  smd6 = &smdtty_apps_riva_ant_data;
  smd7 = &smdtty_data1;
  smd8 = &smdtty_data4;
  smd11 = &smdtty_data11;
  smd21 = &smdtty_data21;
  smd36 = &smdtty_loopback;

  sdhc1 = &sdhc_1; /* SDC1 eMMC slot */
  sdhc2 = &sdhc_2; /* SDC2 SD card slot */
  spi0 = &spi_0; /* SPI0 controller device */
    spi2 = &spi_2;
    spi4 = &spi_4;
  i2c5 = &i2c_5; /* I2c5 cntroller device */
  i2c3 = &i2c_3; /* I2C3 controller */
  i2c1 = &i2c_1; /* I2C1 controller */
  i2c2 = &i2c_2; /* I2C2 NFC qup2 device */
  i2c4 = &i2c_4; /* I2C4 controller device */
  qpic_nand1 = &qnand_1; /* qpic nand controller */
};
```

```
i2c_2: i2c@78b6000 { /* BLSP1 QUP2 */
  compatible = "qcom,i2c-msm-v2";
  #address-cells = <1>;
  #size-cells = <0>;
  reg-names = "qup_phys_addr";
  reg = <0x78b6000 0x1000>;
  interrupt-names = "qup_irq";
  interrupts = <0 96 0>;
  qcom,clk-freq-out = <400000>;
  qcom,clk-freq-in  = <19200000>;
  clock-names = "iface_clk", "core_clk";
  clocks = <&clock_gcc clk_gcc_blsp1_ahb_clk>,
     <&clock_gcc clk_gcc_blsp1_qup2_i2c_apps_clk>;

  pinctrl-names = "i2c_active", "i2c_sleep";
  pinctrl-0 = <&i2c_2_active>;
  pinctrl-1 = <&i2c_2_sleep>;
  qcom,noise-rjct-scl = <0>;
  qcom,noise-rjct-sda = <0>;
  dmas = <&dma_blsp1 6 64 0x20000020 0x20>,
    <&dma_blsp1 7 32 0x20000020 0x20>;
  dma-names = "tx", "rx";
  qcom,master-id = <86>;
};
```

一般情况下，我们不会在 msm8909.dtsi 文件中，对应的 I2C 总线下，添加调试的设备模块信息，而是在项目 dtsi 文件中，加上&i2c_n 这样的开头，然后在其中添加设备信息。这样做可以按照不同项目添加所需的硬件设备，而让代码显得不那么冗余。在对应的项目 dtsi 文件，这儿以 Walabot 为例子，在 QC40A_MSM8909go.dtsi 文件中可以看到有关 gsensor 的设备信息如下：

```
&i2c_1 { /* BLSP1 QUP1 */
  memsic@15 { /* Accelerometer sensor */
    compatible = "memsic,mxc400x";
    reg = < 0x15 >;
    //vdd-supply = <&pm8909_l17>;
    vio-supply = <&pm8909_l6>;
    memsic,dir = "reverse-x-axis-forward";
  };
```

完成以上配置后，我们可以在手机/sys/bus/i2c/devices/1-0015 的路径下，看到我们的设备。如果对应的驱动加载成功，则会在/sys/bus/i2c/drivers/mxc400x/1-0015 路径下，看到驱动注册的一些节点信息。

2. 驱动移植：

驱动的移植一般是根据 FAE 提供的代码，将其配置 OK，编译到我们所需的 kernel 中，根据 MSM8909 的代码架构，我们一般将驱动源代码放到 kernel\msm-3.18\drivers\input\misc 路径下（个人猜测，由于这个平台比较低端，gsensor 还是通过输入子系统的方式上报数据的，所以才放到 input 路径下的）。这儿以 Walabot 项目为例，可以在这个路径下，看到此项目 gsensor 的源代码 mxc400x.c 文件，只要将此路径下的 Makefile 和 Kconfig 文件配置 OK，在对应项目的驱动配置文件（kernel\msm-3.18\arch\arm\configs 路径下的 QC40A_MSM8909go_defconfig 文件）中，加入控制代码编译到 kernel 中的宏，就可以轻松地控制此项目是否要编译此驱动。详细信息见下图：

Makefile：

```
obj-$(CONFIG_SENSORS_BMA2X2)   += bma2x2.o
obj-$(CONFIG_SENSORS_LTR553)   += ltr553.o
obj-$(CONFIG_SENSORS_MXC400X) += mxc400x.o
obj-$(CONFIG_SENSORS_SHT3X) += sht3x.o

ifeq ($(CONFIG_SENSORS_BMA2X2_ENABLE_INT1),y)
  EXTRA_CFLAGS += -DBMA2X2_ENABLE_INT1
endif
```

Kconfig：

```
config SENSORS_MXC400X
  tristate "mxc400X accelerate sensor driver"
  depends on I2C
  help
    Say Y here if you want to enable the MXC400X accelerate sensor
    driver.
    To compile this driver as a module, choose M here: the
    module will be called mxc400x.
```

QC40A_MSM8909go_defconfig：

```
CONFIG_INPUT_MISC=y
# CONFIG_SENSORS_MPU6050 is not set
# CONFIG_SENSORS_AKM8963 is not set
# CONFIG_SENSORS_AP3426 is not set
CONFIG_SENSORS_MXC400X=y
CONFIG_CRC8=y
CONFIG_SENSORS_SHT3X=y
CONFIG_INPUT_UINPUT=y
CONFIG_INPUT_GPIO=m
```

3. 模块调试：

在设备添加好，驱动也注册 OK 的同时，我们还需要保证上层能得到 gsensor 上报的数据，也就涉及上层代码和一些访问文件权限的修改。个人由于对上层不是很了解，在遇到一些上层问题时，在自己努力分析的情况下，大多数还是寻求 FAE 的帮忙。

以下命令可以查看一些基本内容：

1. adb shell
2. getevent -p ：列出所有 input system event，找到加速度的（/dev/input/event2）
3. getevent -l /dev/input/event2（加速度输入子系统事件）：等待数据上报

另外开一个 cmd 窗口，按照以下命令操作：

1. adb shell
2. cd /sys/bus/i2c/drivers/mxc400x/1-0015/sensors/mxc400x-acc
3. echo 1 > enable

操作完以上命令后，在没有问题的情况下，可以看到原来等待数据上报的窗口，现在有很多数据上报。

在此情况下，如果下载一个三方 apk 安装到手机上，不能获取到 gsensor 的数据，说明上层存在问题，可以通过 adb shell logcat > ./Desktop/logcat.log 抓取上层 log 分析，自己不能分析的情况下，可以找 Qcom、FAE 或者同事一起探讨一下。

4. 遇到的问题以及解决方法：

1. 上层 APK 不能获取数据；

通过 logcat 抓取的上层 log 分析，里面看到关于 avc 权限和 enable 节点打不开问题。解决方法：按照 avc 规则，在其对应的文件中，加入所需要的权限；

```
allow hal-server input_device:chr_file r_file_perms;
allow hal-server persist_file:dir search;
allow hal-server sysfs:dir r_dir_perms;
allow hal-server sysfs:file w_file_perms;
```

enable 节点不能 open，通过查找其他 gsensor 有关 enable 节点，依葫芦画瓢，加上此节点所需要的权限。

```
#change mxc400x dev permission
chmod 0666 /sys/class/sensors/mxc400x-acc/enable
chown system system /sys/class/sensors/mxc400x-acc/enable
chmod 0666 /dev/mxc400x
chown system system /dev/mxc400x
```

注：Walabot 项目上，apk 不能获取底层数据的提交记录：http://192.168.1.148:8081/#/c/35/

以上内容仅代表个人的理解，若有不对的地方，还请见谅