嵌入式运动驱动 5.1.1 教程

1.修订历史

2.目标应用

Motion Driver 是一款传感器驱动层程序，其配置简单，利用了 *InvenSense* 的运动传感设备的片上*数字移动处理器*（DMP）的能力。Motion Driver 是嵌入式运动应用软件的一个子集，可以方便地移植到复杂的 MCU 架构。本文档介绍了*运动驱动库*的实际应用。部分教程为兼容 TI 的 MSP430 嵌入式处理器而写，因此，推荐事先熟悉 MSP430 架构。MSP430 仅仅是作为举例平台。**Motion Driver 可以很容易地移植到任何 MCU。**

3.使用需求

3.1 Code Composer Studio 开发环境（仅用于编译 MSP430 例程）
3.2 Motion Driver 源文件
3.3 MotionFit 开发板或相似硬件（仅举例）

4.Motion Driver 简介

Motion Driver 由一套遵循 ANSI 标准 C 语言的 API 构成，这些 API 可用来配置与使用 InvenSense 运动传感器的不同功能，包括 DMP 操作。本教程提供了一个样例工程，程序通过 PC 串口发送经加速度计和陀螺仪数据融合得到的四元数数据，上位机由 python 写就，可在屏幕上显示并旋转一个三维立方体。本驱动支持 InvenSense 的 6 轴和 9 轴设备。

本运动驱动教程包含以下内容：

-如何加载，配置，应用 DMP 功能

-MSP430 的 I2C 驱动示例

-陀螺仪和加速度计自测功能，基于硬件自测文档（自测的具体细节请参考产品寄存器文档）

-加速度计校准及更新硬件寄存器

-陀螺仪校准

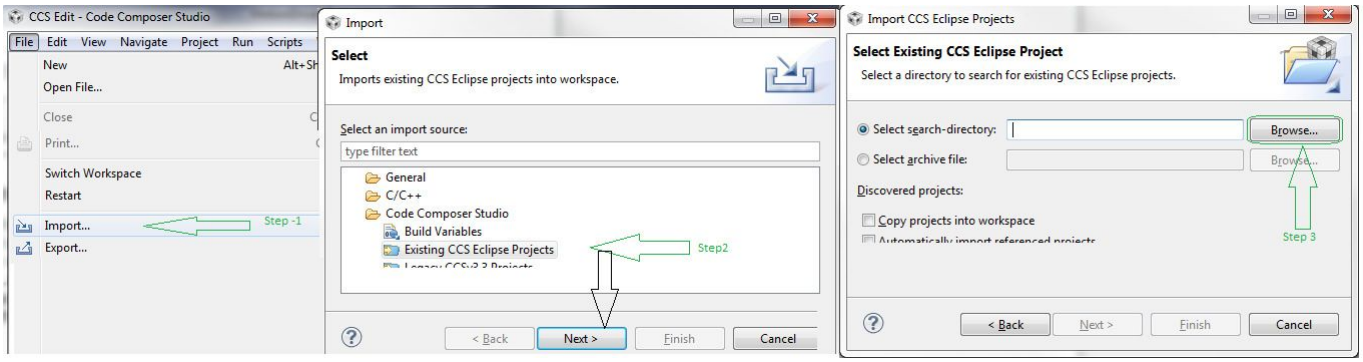
-配置低功耗加速度计运动中断

-可以改变陀螺仪和加速度计的传感器输出数据速率（ODR）

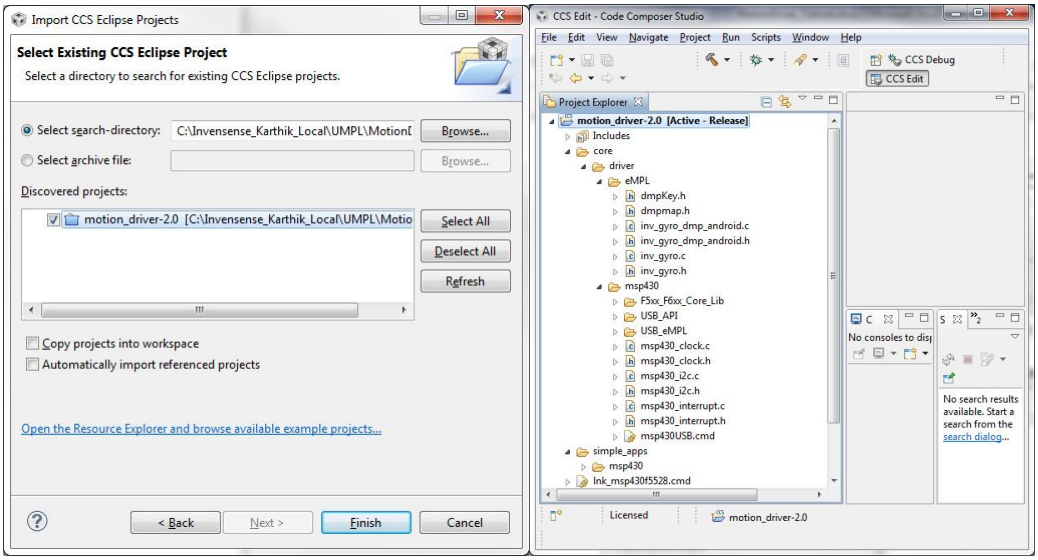
-可以选择 FIFO 中存储哪种数据

5.在 Code Composer Studio 中打开 Motion Driver 工程

1.在 file 选单下选择 import
2.选择现存的 CCS eclipse 工程
3.单击 browse 按钮选择 Motion Driver 文件夹



4.单击完成打开 Motion Driver 工程



5.打开 simple_apps 文件夹下的 motion_driver_test.c 文件。此文件在本教程中将被视为“main”文件。Main 文件包括如何配置 Motion Driver 的样例，并且每一行代码都提供了注释以帮助用户理解代码。

6.更详细地解释 Motion Driver 和功能说明

6.1 DMP

1. DMP 和 DMP 特性：DMP 是 InvenSense MPU(运动处理器)设备独有的硬件特性，它可以通过传感器数值计算四元数，执行设备校准，也包括注入计步器等针对特定应用的特性。DMP 映像（固件）存储在 MPU 的非永久性存储器上，需要在每次芯片上电时更新到 DMP 内，以使能此功能。Motion Driver 的 DMP 映像支持的特性包括：

a. **DMP_FEATURE_LP_QUAT**：DMP 以 200Hz 的频率根据陀螺仪的数据计算三轴四元数，低功耗。

b. **DMP_FEATURE_6X_LP_QUAT**：DMP 以 200Hz 的频率根据陀螺仪和加速度计的数据融合计算四元数并输出，低功耗。

c. **DMP_FEATURE_TAP**：这是一个“轻敲”手势特性，可识别出轻敲事件并分辨其基本的特征，比如单击、双击，或者轻敲的方向。

d. **DMP_FEATURE_ANDROID_ORIENT**：此功能是兼容谷歌 Motion Driver 设备显示方向的实现。此功能包括一个状态机，它计算显示方向。

e. **DMP_PEDOMETER**：计步特性，一直使能，在 MPU 上电后即在 DMP 中运行。Motion Driver 库可以复位计步器步数值，查询步行时间，以及返回步数。为改善精度、避免误判，在更新步数前会有 7 步的延时。此特性随 DMP 的使能一直使能。

f. **DMP_FEATURE_GYRO_CAL**：此特性将在每次设备静止超过 8 秒时，校准陀螺仪零偏。

g. **DMP_FEATURE_SEND_RAW_ACCEL**：将加速度计的 raw 轴（偏航）的数据放入 FIFO，此数据基于芯片坐标系。

h. **DMP_FEATURE_SEND_RAW_GYRO**：将陀螺仪的 raw 轴（偏航）的数据放入 FIFO，此数据基于芯片坐标系。

i. **DMP_FEATURE_SEND_CAL_GYRO**：将陀螺仪的校准后的数据放入 FIFO。不可与 DMP_FEATURE_SEND_RAW_GYRO 结合使用。输出不基于芯片框架，而是基于设备框架或者身体框架。

2.加载并使能 DMP 特性的步骤：更新映像和初始化 DMP 特性的步骤已在 Motion Driver 库提供的参考应用工程里高亮。这些步骤包括：

a.推送 DMP 映像到 MPU 内存：提供了 dmp_load_motion_driver_firmware 函数帮助实现此任务

b.推送陀螺仪和加速度计的方向矩阵到 DMP：可以发送方向矩阵到 DMP 来转换输出为参考板子或设备坐标系的输出。dmp_set_orientation () 函数用来更新方向矩阵。

c.DMP 回调函数：一些 DMP 特性注册回调函数向主程序传递警讯，这些回调函数在诸如方向准备、检测到轻触等 DMP 特性下触发。在 Motion Driver 里提供了一些函数作为例子，比如方向改变特性，将触发 dmp_register_android_orient_cb(android_orient_cb);。注意注册一个回调函数并不意味着默认打开了该特性。

d.使能 DMP 特性：提供了函数 dmp_enable_feature ();

e.数据速率：FIFO 速率定义了 DMP 的输出速率，可以通过函数 mpu_set_fifo_rate (input) 更新。

6.2 与 InvenSence 设备通讯的 I2C 驱动

1. Motion Driver 库以源代码的方式提供了一个 MSP430F5528 平台的 I2C 驱动实例。使用 msp430_i2c_write 和 msp430_i2c_read 函数进行基本的数据读写， msp430_i2c_enable 和 msp430_i2c_disable 函数用以使能/失能 MSP430F5528 的 I2C 通讯。这些函数需要的形参包括*从地址*、*寄存器地址*、*长度和数据*。在 msp430_i2c.c 文件中有更多详细的信息。

6.3 陀螺仪和传感器自测（self-test）函数调用

MPU 的陀螺仪和加速度计自测特性允许用户测试陀螺仪和加速度计的机械和电子部分。当自测激活，板上电路将“激励”相应的传感器。此激励将移动陀螺仪的测试单元一定距离，等效于科里奥利力（地球自偏转）；模拟外部力施加在加速度计上。测试结果改变传感器的输出，映射在输出信号上。输出信号通常用于与自测寄存器比较，观测“自测响应”。“自测相应”（STR）定义如下：

SelfTest Response=Sensor Output with SelfTest Enable- Sensor Output with Selftest Disabled

（STR 自测响应=自测使能时传感器的输出-自测失能时传感器的输出）

此自测响应与工厂设定的自测响应值比较，用于判定该部分是否通过了自测试：

Change from Factory Trim of the Self-Test Response(%)=(Str-FT)/FT

Motion Driver 提供了 int mpu_run_self_test(long* gyro, long* accel) API，来执行陀螺仪和加速度计的自测。如果自测成功执行将返回 0。从 mpu_run_self_test 获得的陀螺仪值应根据陀螺仪敏感度设置做相应的比例转换。陀螺仪敏感度参数可通过调用 mpu_get_gyro_sens(float* sens)获得。按比例缩减陀螺仪和加速度计的 16 位有符号格式（Q16）的数据，存储最新获得的陀螺仪基准值。

执行加速度计自测用以确定加速度计功能正常，该功能通过 mpu_self_test()函数中调用另

一个函数 accelerate_self_test(accel,accel_st)实现。**accel_st** 参数参考标准加速度计基准值，该值可通过调用 get_st_biases 从 MPU 寄存器中获得。
motion_driver_test.c 中有关于 run_self_test 函数的更多细节。参考以下函数可获得更多信息：

- mpu_run_self_test
- get_st_biases
- accel_self_test

6.4 加速度计校准

将板子放在平坦表面上，保持静止，加速度计校准将通过对照当前加速度值（如有必要）更新基准值。加速度计校准执行的函数为 accel_self_test(accel,accel_st)，该函数包含于 **inv_mpu.c**。该函数检索 MPU 寄存器获得加速度计的基准偏差（函数调用 get_st_biases 返回基准偏差）和当前加速度计读取值，计算两值之差与 **inv_mpu.c** 中 **test_s** 结构体的一个实例所指定的最大、最小 g 值相比较。

```
const struct test_s test = {
    .gyro_sens      = 32768/250,
    .accel_sens     = 32768/16,
    .reg_rate_div   = 0,    /* 1kHz. */
    .reg_lpf        = 1,    /* 188Hz. */
    .reg_mpu_fsr    = 0,    /* 250dps. */
    .reg_accel_fsr  = 0x18, /* 16g. */
    .wait_ms        = 50,
    .packet_thresh  = 5,    /* 5% */
    .min_dps        = 10.f,
    .max_dps        = 105.f,
    .max_gyro_var   = 0.14f,
    .min_g          = 0.3f,
    .max_g          = 0.95f,
    .max_accel_var  = 0.14f};
```

调用 mpu_set_accel_bias 函数，更新 MPU 硬件寄存器中的加速度计标准偏移量。注意：陀螺仪偏移量更新到 DMP 存储器，加速度计偏移量更新到硬件寄存器。参考 mpu_set_accel_bias 获得更多细节。

6.5 陀螺仪校准

DMP 提供了一种基于设备不移动状态的方法来校准陀螺仪偏移量。此特性可通过选择 DMP 特性中的 DMP_FEATURE_GYRO_CAL 使能。当此特性使能，并且板子保持不移动超过 8 秒，陀螺仪将自动校准。

6.6 低功耗加速度计运动中断

此段揭示了驱动中关于实现低功耗（LP）的加速度计中断模式的部分，可用于当无运动时休眠主进程直到检测到运动。函数
int mpu_lp_motion_interrupt (unsigned short thresh, unsigned char time, unsigned char lpa_freq) 配置 LP 加速度中断模式的三个参数（在 Embedded Motion Driver API 文档中有进一步描述），包括阈值、时间和 LPA 频率。
在此模式下，设备将以固定的频率持续采样加速度计，直到检测到持续一段时间的高于阈值的信号。如果选择的时间周期比 Ipa_freq 定义的采样周期值小，LP 加速度中断将在采集到高于阈值信号后的第一次采样时触发。参照产品规格文档（DS）的低功耗加速度中断章节可获得更多细节。

6.7 能够改变传感器加速度计和陀螺仪的 ODR

InvenSense MPU 提供了可编程的陀螺仪、加速度计数据输出速率范围（ODR）。该范围可通过写值到 SMPLRT_DIV 配置，陀螺仪的输出数据速率计算公式如下：

Sample Rate = Gyroscope Output Rate /(1+SMPLRT_DIV)

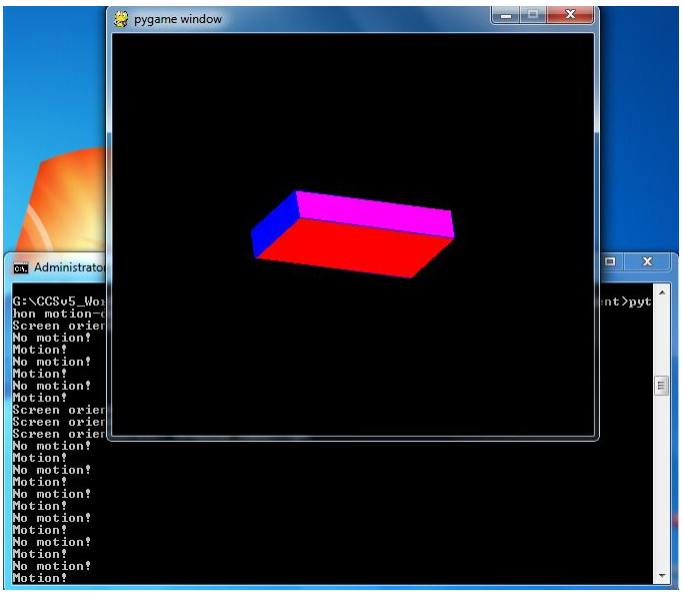
Motion Driver 提供了一个简单的方法配置陀螺仪的 ODR——调用函数 mpu_set_sample_rate。不过，当 DMP 打开，陀螺仪就预设为 200Hz 采样率，并不需要调用 mpu_set_sample_rate。当 DMP 关闭，最大值可以设置为最高 8Khz，取决于 MPU 设备特性。当 DLPF 失能（DLPF_CFG=0 or 7）最大陀螺仪输出速率为 8Khz，当 DLPF 使能时为 1Khz。最大加速度计输出速率为 1Khz。当采样率高于 1KHz 时，加速度计的同一次采样数据可能会在 FIFO、DMP 和传感器寄存器中输出超过一次。

6.8 能够通过调用函数选择填充到 FIFO 中的数据

提供了片内 FIFO 来队列化传感器寄存器值，降低了 MCU 读取速率的需求。只有加速度计和陀螺仪传感器数据可以被推到 FIFO 缓冲区。每个轴陀螺仪数据可以存储在独立 FIFO(如果需要的话)。加速度计输出只能用于同时保存所有三个轴值。Motion Driver 提供了函数 int mpu_configure_fifo(unsigned char sensors) 来配置 FIFO 缓冲区。sensor 的值包括如下标志： INV_X_GYRO, INV_Y_GYRO, INV_Z_GYRO, INV_XYZ_GYRO, INV_XYZ_ACCEL.

7.使用 MotionFit SDK 和 Python 脚本测试 Motion Driver

- 编译工程加载到 SDK
- 连接 USB，指向.inf 文件（core\driver\msp430\USB_eMPL*.inf）安装 USB CDC 驱动
- 参考 APPENDIX_A 安装 python
- 获得 SDK 连接的串口号。关于获得串口号的帮助，参考 APPENDIX_B
- 打开命令行界面，指向 motion-driver-client.py 存储的文件夹，例如 C:\MotionDriver\embedded_motiondriver-Rel-TI_MSP430F5528-MPU6050-V5_1_1-20 1212-14\simple_apps\msp430\motion-driver-client
- 在命令行中输入字符 **motion-driver-client.py 32** 运行 motion-driver-client.py
- 32 是步骤#4 中获得的串口号，将其改变为你在步骤#4 中获得的串口号（参见 Appendix B）
- 应用成功运行，显示如下窗口：



- Motion Driver 提供了一个命令集，从 PC 发送指令来执行相应的任务，比如自测等。

下面是命令列表，每个命令在发送到 PC 时应加前缀“inv”，例如命令“a”应为“inva”，焦点中的窗口将旋转方块。

- a：触发传送加速度计值到控制台的状态
- g: 触发传送陀螺仪值到控制台的状态
- q：触发传送四元数值到控制台的状态
- t: 运行自测
- f: 触发 DMP
- p: 读取当前的 pedometer 值
- x: 复位系统
- v: 触发 LP（低功耗）四元数
- m: 测试运动中断
- 1：如果 DMP 打开设置 DMP_FIFO_rate 为 10Hz，如果 DMP 关闭则设置 gyro_sample_rate 为 10Hz
- 2：如果 DMP 打开设置 DMP_FIFO_rate 为 20Hz，如果 DMP 关闭则设置 gyro_sample_rate 为 20Hz
- 3：分别为 40Hz
- 4：分别为 50Hz
- 5：分别为 100Hz
- 6：分别为 200Hz
- 7：重设计步器计数值和步行时间为 0
- 8：触发当 DMP 不打开的时候仅运行加速度计时
- 9：触发当 DMP 不打开的时候仅运行陀螺仪
- ,：设置硬件仅在姿态事件中断。此特性对于保持 MCU 睡眠，直到 DMP 检测到一次敲击或者方向事件很有用。
- .: 设置硬件周期性中断。

8.通过 Motion Driver 获得罗盘数据

如果使用集成电子罗盘的 InvenSense MPU-9150 或者一个 6 轴 MPU 通过第二条 I2C 总线连接一个辅助的电子罗盘，motion driver 有能力从电子罗盘中拉取 raw（偏航）数据。Motion driver 函数 mpu_set_sensors() 可以用来配置电子罗盘，同样也可以用 INV_XYZ_COMPASS。当其配置好后，用函数 mpu_set_conpass_sample_rate()设置电子罗盘采样率，最大 100Hz。mpu_get_compass_reg()功能将允许获得芯片层面的电子罗盘数据。

9.罗盘融合和校准

本节提供关于如何集成和校准电子罗盘一般性指导和参考，以实现 9 轴传感器融合。Motion Driver 源代码不包括或支持在本节点出的任何电子罗盘校准或融合参考。本节讨论的电子罗盘校准和集成参考自 TI 和 MEMSense 提供的应用笔记和源代码。

磁场强度测量受地磁场和当地铁磁材料的影响。考虑理想情况下,磁场沿着硬件组件磁力仪应该形成一个球体，原点应该在(0,0,0)。然而，只是不考虑外部磁场影响的情况。如果外部影响来自于预设的磁场，或者“硬铁”，将会导致球体的原点偏移（$\Delta x, \Delta y, \Delta z$），这可以通过地磁场数据类似基准偏移量校准的操作来校准。“软铁”效应是由于材料弯曲和扭曲了当地磁场,这将导致角度精度误差。为得到正确的电子罗盘值，软铁影响也应消除。

以下链接解释了罗盘指南针校准和集成并给出了源代码。此应用笔记详细解释了硬件如何使用。

- http://www.memsense.com/docs/MTD-0802_1.2_Magnetometer_Calibration.pdf
- http://www.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=slaa518a&fileType

=pdf

10.SPI 驱动实现

略

11.附录

11.1 附录 A

略

11.2 附录 B

略

12.参考文档

[1] Embedded Motion Driver V5.1.1 APIs Specification, Doc# SW-EMD-REL-5.1.1