Motion Driver 6.12-移植指南

2018年8月11日

22:59

4 Motion Driver 6.12 Features 
This is a quick overview on the Mth12 features. 
DMP Features: 
0 3/6 Axis Low Power Quaternions 
o Tap, Orientation, and Pedometer Gesture Detections 
MPL Algorithms: 
Run Time Gyro Calibration 
Run Time Gyro Temperature 
Run Time Compass Calibration 
Run Time Magnetic Disturbance Rejection 
0 3/6/9 Axis Sensor F usion 
Hardware Features: 
o Factory Calibration 
o Factory Self Test 
Saving and Loading Sensor States 
Low Power Accel Mcx:ie 
Low Power Motion Interrupt Mode 
Register Dump 

# 1、选择MCU

对于每个嵌入式系统，功能和性能取决于所选的MCU。成本，低功耗，速度，工具链和处理都是需要考虑的因素。对于MPU设备，如果您计划使用InvenSense Motion Driver 6.12软件，则需要考虑以下事项。

* **闪存和RAM大小**：闪存和RAM大小取决于代码优化，编译器，您要使用的功能以及系统中的其他组件。一般而言，MD6.12要求您需要保留以下数量的Flash和RAM。请记住，这仅适用于运动驱动程序，而不适用于其他可能的功能。

o 16位MCU - 128K和12K

o 32位MCU - 68K和10K（无优化，64K和8K优化）

由于依赖大小非常依赖于编译器和编译器设置，客户应该花时间确定闪存的大小他们的应用程序需要RAM。

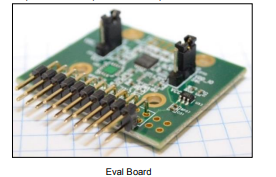
* **长数学支持**：MPL库需要支持长（64位）数学。您需要确保是否使用MPL库，您的工具链可以支持此功能。通常8051 MCU不支持这种数学计算。如果工具链不支持长时间的数学运算，您仍然可以使用DMP来获得6轴融合。

* **中断**：MPU设备可以为低功耗手势识别或数据就绪中断提供各种功能的中断。虽然系统不需要使用MPU中断，但如果您打算使用它，则必须保留具有唤醒功能的GPIO引脚。

* **采样率**：传感器融合需要MCU的大量计算能力。这会影响每个样本的处理量并限制采样率。例如，如果MCU正在进行完整的9轴融合，则带有运动驱动器的TI 16位MSP430应限制为100Hz采样率。任何超过100Hz采样率的MSP430都会开始丢失数据。如果系统中不需要其他大型计算功能，则高端32位MCU通常可以**实现200Hz传感器融合**。如果将处理卸载到DMP上，则可以提高此采样率。

# 2、连接硬件

选择MCU后，您很可能会拥有一个MCU评估套件或您自己的PCB板。要将MPU器件连接到MCU板进行评估，可以通过InvenSense.com获得InvenSense MPU评估板。MPU6050，MPU6500，MPU9150和MPU9250均可提供。



您需要将以下引脚从评估板连接到MCU板

VDD and VDD_IO (pin 23): Depending on the MPIJ device this would a 3V or a 1.8V (see device 
spec) 
SDA and SCL (pins 20, 22): 12C pins 
GND (pins 15 or 17): Connect to ground 
INT (pin 7): Connect to GPIO for interrupts (optional but needed f using Invensense software) 

InvenSense评估板的引脚输出是相同的

要确认您的硬件设置以及基本的I2C功能，请首先阅读MPU设备的**who am i**寄存器并确认您获得了正确的设备ID。对于MPU系列，I2C地址为0x68，而不同器件的器件ID都不同，请查看规格。硬件确认后，我们可以继续进行软件集成。

# 3、Motion Driver 6.12固件包

Motion Driver 6.12发行版固件包含以下文件夹：

* **core \ driver**： 此文件夹包含MPU设备的InvenSense驱动程序层以及MCU特定的驱动程序
* **simple\_apps\ msp430 \ mllite\_test.c or src \ main.c**： 项目应用程序的主要功能和主循环。客户可以使用此代码作为参考，将驱动程序功能集成到他们的项目中。
* **core\ mllite**： 此文件夹包含MPL数据处理功能，用于存储接收的传感器数据并处理数据
* **core\ mpl**： 包含InvenSense专有MPL库 - 包含用于传感器融合和运行时校准的高级算法的库。
* **core\ eMPL-hal**：此文件夹包含提供传感器数据转换的文件，例如欧拉角。

# 4、集成Motion Driver 6.12

嵌入式MD6.12包含以下需要集成在目标硬件平台上的组件：

1) Driver

2) Motion Processing Libraries

3) Sample HAL

下图显示了Motion Driver 6.12的体系结构以及将MD6.12成功移植到目标平台所要执行的任务。

MPL 
MAIN 
Application-specific: 
1. Data output (and i 
t). 
2. Client handling, 
3. Power management. 
Driver 
APIs 
Driver 
Layer 
Platform- ecific: 
1. 12C read/write. 
2. System clock (ms). 
3. Hardware interrupts. 
4. Log outputs. 
MPL 
NO needed! 

### 1、Invensense MD6.12驱动程序层（core \ driver \ eMPL）由这些文件组成

* **inv\_mpu.c** - 可以轻松移植到不同嵌入式平台的驱动程序。
* **inv\_mpu.h** - 包含InvenSense驱动程序的结构和原型。
* **inv\_mpu\_dmp\_motion\_driver.c** - 包含dmp映像的驱动程序以及用于加载和配置DMP的API。
* **inv\_mpu\_dmp\_motion\_driver.h** - 包含DMP功能的原型和定义
* **dmpKey.h** - 包含DMP功能的DMP内存位置定义
* **dmpmap.h** - 包含DMP内存位置的定义

用户需要提供以下API以支持I2C读/写功能，系统时钟访问，硬件中断回调以及与要移植MD6.12的平台相对应的日志记录。

这些函数需要在inv\_mpu.c和inv\_mpu\_dmp\_motion\_driver.c中定义。以下是MSP430平台的示例。

#define i2c_write 
#define i2c read 
#define delay_ms 
#define get_ms 
#define log_i 
#define log_e 
msp430 i2c_write 
msp430_i2c_read 
msp430 delay_ms 
MPL_LOGI 
MPL_LOGE 

**i2c\_write和i2c\_read**：这需要链接到i2c驱动程序。此函数将接受4个参数，然后执行i2c事务

unsigned char slave_addr 
unsigned char reg_addr 
unsigned char length 
unsigned char *data 

**delay\_ms**：此函数将接受一个无符号长参数，它将作为系统的延迟（以毫秒为单位）

**get\_ms**： get\_ms主要用于获取当前时间戳。时间戳通常是无符号长整数，以毫秒为单位。该功能主要用于罗盘调度器以及传感器融合数据的附加信息。

**log\_i和log\_e**：MPL消息传递系统，可以记录信息或错误消息。当前实现对消息进行分组并通过USB或UART将其发送出去供python客户端接收。日志记录代码位于log\_msp430.c或log\_stm32l.c文件中。客户可以根据自己的喜好更改传输方法和数据包。

### 2. MPL Library是专有的InvenSense Motion Apps算法的核心，由Mllite和mpl目录组成。MPL不需要移植。您可能需要包含系统特定的头文件，以支持mllite包中的memcpy，memset，...等函数调用。MD6.12软件包将包含预编译的MPL库，一个用于TI MSP430平台，另一个用于ARM核心平台。ARM库有一个通用库，可以链接到任何ARM MCU。它还有一些针对M3和M4的预编译，以便进行更好的优化。

### 3. eMPL-HAL目录包含用于从MPL库获取各种数据的API。您可以获得的数据来自以下APIs

int *data, int8_t *accuracy, 
*timestamp) ; 
int *data, int8_t *accuracy, 
*timestamp) ; 
int *data, int8_t *accuracy, 
*timestamp) ; 
int *data, int8_t *accuracy, 
*timestamp) ; 
int *data, int8_t *accuracy, 
*timestamp) ; 
int *data, int8_t *accuracy, 
*timestamp) ; 
int *data, int8_t *accuracy, 
*timestamp) ; 
int *values, int8_t *accuracy, 
inv _ time t * timestamp) 

4. main.c或mllite\_test.c包含一个非常具体的应用程序

* 来自MPU设备的传感器数据处理句柄
* 处理来自客户端的输入请求
* 电源管理
* 初始化MPL库，DMP和硬件
* 处理中断

# 5、初始化API

在通电时，MPU设备将在其默认状态下提供传感器数据。inv\_mpu.c提供了有关如何使用一些基本配置初始化MPU设备的参考API，例如打开传感器电源并设置比例范围和采样率

int 
int 
int 
int 
int 
int 
int 
int 
short fsr) 
char fsr) 
short Ipf) 
short rate) 
short rate) 
char sensors) 
char sensors) 

# 

# 6、方向矩阵

应用程序还需要定义MPU设备的方向矩阵和平台上存在的第三方罗盘。方向矩阵将物理硬件传感器轴重新配置为设备坐标。错误的配置会使传感器数据的结果不准确。有关如何定义矩阵的更多信息，请参考Orientation Matrix Transformation chart.pdf。

矩阵将被推入MPL库和DMP以进行融合计算。

struct platform data_s { 
signed char orientation[9]; 
P The sensors can be mounted onto the board in any orientation. The mounting 
• matrix seen below tells the MPL how to rotate the raw data from the 
• driver(s). 
static struct platform_data_s gyro_pdata = { 
-orientation {-1, O, O, 
0,-1, o, 
static struct platform data_s compass_pdata { 
#ifdef 
-orientation , 
0,-1} 
0,-1} 
#else 
.orientation 
#endif 
O, 
O, 

# 7、中断处理

MPU器件有一个中断输出引脚。可以将中断编程为生成以下中的任意一个：

* FIFO输出速率
* 生成DMP

通常当FIFO中有新的传感器数据可用时，我们会生成中断。DMP还可以被编程为在检测到手势时生成中断。如果您使用的是MD6.12参考示例，则在生成传感器数据就绪中断时，中断例程将全局标志new\_gyro设置为1.

在主循环中，它将知道有一组新的传感器数据要处理。

以下是与中断相关的API列表：

int char mode) 
• static int char enable) 

# 8、DMP - 数字运动处理器™

MPU-9150，MPU-6050，MPU-9250和MPU-6500均具有嵌入式数字运动处理器™（DMP）硬件加速器引擎。DMP与嵌入式FIFO一起从主机应用处理器卸载高频运动算法计算，减少中断和主机MIPS以提高整体系统性能。所有相关的DMP API和固件都可以在inv\_mpu\_dmp\_motion\_driver，dmpKey.h和dmpMap.h中找到。

# 9、DMP Intialization

DMP固件代码是在结构中找到的3kB映像

C:\2C698EE5\D1EE5E74-EC2F-45DD-ACB2-71378C34E1B6.files\image011.png

该映像需要下载到DMP存储库中。下载后需要提供起始地址，然后需要打开DMP状态。与DMP初始化相关的API如下：

计算机生成了可选文字:
·intdmp_load_motion_driver_fir，Nare(void）
·intdmp_load_motion_driver_fir，Nare(void）
·intdmp_set_fifo_rate（unsignedshortrate)
·intmpu_set_dmp_state（unsignedCharenable)

DMP初始化的MD6.12示例可以在主循环之前的main函数中找到。

# 10、DMP特征

DMP具有许多功能，详见“功能指南”。可以动态启用和禁用这些功能。主要的API是

C:\2C698EE5\D1EE5E74-EC2F-45DD-ACB2-71378C34E1B6.files\image013.png

此功能将掩码和索引带入DMP固件中的正确内存地址，以启用和禁用该功能。特征如下：

计算机生成了可选文字:
#defineDMP
#defineDMP
#defineDMP
#defineDMP
#defineDMP
#defineDMP
#defineDMP
#defineDMP
#defineDMP
FEATURE
TAP
FEATIJRE
ANDROID_ORIENT
FEATIJRE
LPQUAT
FEATIJRE
PEDOMETER
FEATURE_6X_LP_QUAT
FEATIJRE
GYRO_CAL
FEATIJRE
SENDR众ACCEL
FEATIJRE
SEND_RAW_GYRO
FEATIJRE
SEND_CAL_GYRO
x1）
x2）
x4）
x8）
(øxØIØ)
xe2劬
x249）
x289）
xl）

对于Tap和Orientation数据解析，MD6.12驱动程序定义了2个回调函数，它们将处理解析并将其记录到python客户端。回调将需要定义MD6.12驱动程序。相关的API是

计算机生成了可选文字:
intdmp_register_tap_cb(void(*func)(unsignedChar,unsignedChar))
intdmp_register_android_orient_cb(void(*func)(unsignedChar))
staticintdecode_gesture(unsignedChar*gesture)
staticvoidtap_cb(unsignedChardirection,unsignedCharcount)
staticvoidandroid_orient_cb(unsignedCharorientation）

Tap还有一些可配置的设置，例如阈值。这些API在inv\_mpu\_dmp\_motion\_driver中可用

# 11、DMP FIFO Output

当启用特定功能（如分接头或传感器数据）时，DMP仅写入FIFO。MD6.12驱动程序将等待DMP生成中断，然后读取FIFO的内容。FIFO格式取决于启用哪些DMP功能.DMP FIFO输出格式可在API函数中看到。

计算机生成了可选文字:
·intdmp_readfifo(short*gyro,Short庫a（（el，long*quat,
unsignedlong*timestamp,short*sensors,unsignedChar*more)；

# 12、InvenSense硬件自检

硬件自检是一项可选的工厂生产线测试，客户可以在其生产线上作为合格/不合格测试。HWST算法将测试MEMS传感器并通过内部移动和测量运动并将输出与其寄存器中保存的Invensense数据进行比较来确认工作功能。有关更多详细信息，请查看产品规格。

MD6.12代码提供了有关如何运行HWST及其输出的示例代码。硬件自测试可以在没有与MPL进行任何交互的情况下运行，因为它完全本地化在驱动程序中。运行完整自检的API是

C:\2C698EE5\D1EE5E74-EC2F-45DD-ACB2-71378C34E1B6.files\image017.png

MD6.12将自检和工厂校准捆绑在一起，因为传感器偏移是通过正常的自检程序计算出来的。但是，如果客户希望，客户可以将校准和自检分开。

与MPU6500 / MPU9250相比，MPU6050 / MPU9150具有不同的自检算法。API返回传感器每个轴的状态以及用于校准的加速度和陀螺仪偏置

计算机生成了可选文字:
intmpu_run_65øeself_test(long*gyro,long*accel,unsignedChardebug)
·intmpu_runself_test（long*gyro,long*accel)

自检功能的参数如下所示：

计算机生成了可选文字:
Parameter
type
output
Input
ParameterName
result
accel
gyro
debug
ThefunctionreturnstheresultOftheself-testas
showninthetable《OW
Returnstheaccelbias.
Returnsthegyrobias.
Extralogsfortheselftest.Defaultis0

'结果'的返回值定义如下，带'1'

计算机生成了可选文字:
Value
0×01
0×02
0×04
SensorStatus
GyroSensorStatus
AccelSensorStatus
CompassSensorstatus

如果返回的值不是0x07，则表示特定传感器发生故障。下面是从python脚本启动的自测试传递输出的示例。

计算机生成了可选文字:
pythoneMPL—client．py？9
Passed!
accel:．g194gg121．g152
一3“284g·789一g·4

如果自检失败，如果从python脚本启动自检命令，则失败的传感器将显示在python脚本窗口中。

# 13、校准数据和存储

校准数据包含描述MPU陀螺仪，加速度计和罗盘的固有偏差和温度相关行为的信息。在MPL执行期间使用此数据以提高MPL返回的结果的准确性。校准数据可能会随着时间，温度和环境的变化而缓慢变化，因此Invensense提供了几种使用中的传感器校准算法，可在整个生命周期内不断校准传感器。“功能指南”中介绍了详细信息。建议在工厂生产线上校准MPU传感器加速度和陀螺仪，如果使用MPL库打开使用中的算法。

## 13.1 工厂线校准

自测功能返回的加速度和陀螺仪偏差可用于工厂校准，可由HAL保存并用于校准传感器的性能。偏置可以推送到HW Offset寄存器或MPL库。

默认情况下，MD6.12将偏差推入MPL库，并让融合引擎应用偏差数据。但是，客户可以通过定义来使用硬件偏移寄存器。

C:\2C698EE5\D1EE5E74-EC2F-45DD-ACB2-71378C34E1B6.files\image022.png

在main.c.不同之处在于，如果使用HW偏移寄存器，MEMS数据将在被推入传感器数据寄存器之前自动进行调整。要更好地了解硬件偏移寄存器，请参阅应用笔记“MPU硬件偏移寄存器”。重要的是，当您进行工厂线校准时，设备需要处于稳定且无振动的环境中，物理Accel Z +朝上或朝下，这将是MPU IC需要向上或向下指向的面。

## 13.2 保存和加载校准数据

MPL不会自动生成，加载或存储校准数据。计算并应用偏差后，一旦设备断电，它就会丢失。因此，InvenSense提供了有关如何从内存位置保存和加载校准数据的API示例。请查看功能

计算机生成了可选文字:
·inv_err，ortinvsave_mpl_states(unsignedChar*data,Size_tSz)
·inv_error_tinvload_mpl_states(constunsignedChar*data,size_tlength）

客户可以使用这些功能作为如何将其保存到设备内存中的示例。出厂校准后和断电前应保存校准。开机时需要重新加载。

# 14、集成MPL库

MPL库是包含传感器融合引擎的预编译库。移植MD6.12时，库需要与集成商系统兼容。MD6.12附带2个库。

* TI MSP430 - 使用Code Composer编译。应与所有MSP430产品系列兼容
* ARM - MD6.12附带了许多ARM预编译库。有特定的IAR，Keil和GNU 4.9.3编译库。每个编译器使用M0，M0 +，M3，M4和M4F的特定设置编译库。

链接库后，代码将需要启用库及其功能。库初始化可以在主循环之前的main函数中找到。功能指南中介绍了这些功能。以下是相关的API

计算机生成了可选文字:
Inv
Inv
Inv
Inv
Inv
Inv
Inv
Inv
Inv
error
error
error
error
error
error
error
error
error
_tinvinit_mpl(void)
_tinv
enable_quaternion(void）//enable6-axis
_tinvenable_9x_sensor_fusion(void)//enable9-axisfusion
_tinv
enablefast_nomot(void）//gyroin-usecalibration
_tinv
enable_gyro_tc(void)//gyrotemperaturecompensation
_tinv
enable_vector_compass_cal(void)//compasscalibration
_tinv
enable_magnetic_disturbance(void)//magneticdisturbance
_tinv
_enab1e_eMPL_outputS(VOid）
_tinv_start_mpl(void)

# 15、MPU6500 / MPU9250的低功耗加速模式和运动中断模式

LPA模式和运动中断模式类似，可以启用设备只需要加速数据。MPU6050 / MPU9150不支持此功能。此功能要求禁用DMP，FIFO和陀螺仪。然后，它将按照用户请求的指定速率将加速器唤醒的周期进行休眠循环。不同的LPA速率从1.25Hz到640hz。速率越低，消耗的功率越低。总功率最低，约为10uA。客户还可以将设备置于运动中断模式。在此模式下，设备将处于LPA模式，如果加速数据超过某个阈值，则会产生中断。如果没有动作并且客户想要在检测到移动之前睡眠设备，这尤其有用。

相关的API是 -

计算机生成了可选文字:
intmpu_lp_accel_mode（unsignedshortrate)
·intmpu_lp_motioninterrupt(unsignedshortthresh,unsignedChartime,
unsignedShortIpa_freq)

# 16、编译器特定设置

要编译不同的部件（MPU6050，MPU9150，MPU6500和MPU9250），您需要设置编译器标志

计算机生成了可选文字:
，PropertiesforeMPL-5D
typefiltertot
Resource
《0General
〔0Build
M430Compiler
M430Linker
C/C·Build
BuildVariables
D巧（OVOptions
Environment
Logging
Settings
C/C·General
Debug
ProjectReferences
RefactoringHto
Run/DebugSettings
Configuration:Debug[Activel
T“巧地gs户Build印
〔Tool-chainSettings
0MSP430Compiler
ProcessorOptions
〔Optimization
DebugOptions
〔IncludeOptions
AdvancedOptions
BuildArtifact
生8貊ar．Errorr丐
Pre-defineNAME(--define,-D)
M」P4芟F55
CONFIGINTERFACEUSB
MPLLOGNDEBUG=I
MP1
EMPL
EMPLTARGETMSP4
USEDMP
f'AdvancedDebugOptions
LanguageOptions
ParserPreprocessingOptions
PredefinedSymbols
[Diagnostic0@0n5

需要的默认符号是：

计算机生成了可选文字:
MPLLOGNDEBUG=I
MPU9150orMPU6050orMPU6500orMPU9250
EMPL
USEDMP
EMPL_TARGETMSP430oritsequivalent

**设置部件后，编译器将针对该特定部件及其功能进行编译。**

评估板

CM 
HEADER ιοΧ.2. M". 2Ε4π•π-α2.54η-π 
μυχ E'.'22 
ΤΡ2 EVB2 
DROY ΤΡΟ EVE2 
FSYNC ΤΡΙ EVB2 
ΤΡΙ Ε'ι,'Β2 
REGOU'T EVE2 