Freescale Semiconductor
Application Note

Document Number: AN4601
Rev 0, 12/2012

Smart Power Management for Mobile GPS using Xtrinsic MMA8452Q

By Kevin Jia, Applications Engineer

1 Introduction

The MMA8452Q device can be used to detect whether a car is moving or has stopped. By combining an accelerometer sensor's auto-wake/sleep and transient detection function, this application note shows you how to use the MMA8452Q accelerometer to perform smart power management for a handheld GPS unit when used in a car, without any inputs from the people in the car.

Imagine:

- You enter a car, pull out your GPS unit and set an address.
- When the car starts to move, the accelerometer keeps the GPS in working mode.
- Eventually, you arrive where you are going.
- When the car stops, the GPS unit automatically enters sleep mode to save battery power.

Contents

2 Design Idea

The GPS module has the MMA8452Q device on its board. The accelerometer is used to detect the car's motion (moving or stopped):

- When GPS is in sleep mode and the car starts moving, the accelerometer detects the motion and sends a wakeup interrupt to the MCU, to wake up the entire GPS system.
- When the GPS is active (not asleep) and the car stops moving, the accelerometer detects the stop and sends a sleep interrupt to the MCU, to put the entire GPS system into sleep mode.

2.1 Implementation

The MMA8452Q provides many embedded DSP functions, but for our design, we only need auto-wake/sleep detection and transient detection.

While the auto-wake/sleep function can make the GPS module enter sleep mode, the accelerometer can also fall into sleep mode. The possible interrupts that can wake the accelerometer from sleep mode are:

- Tap detection
- Orientation detection
- Motion/freefall detection
- Transient detection

We selected transient detection as the wakeup source, because the transient detection function has a high pass filter (HPF) option.

2.2 Acceleration vectors

Consider a car's movement or stop on a slope—there is a static component of G (Gx) that points in the forward or back direction of the vehicle. See Figure 1.

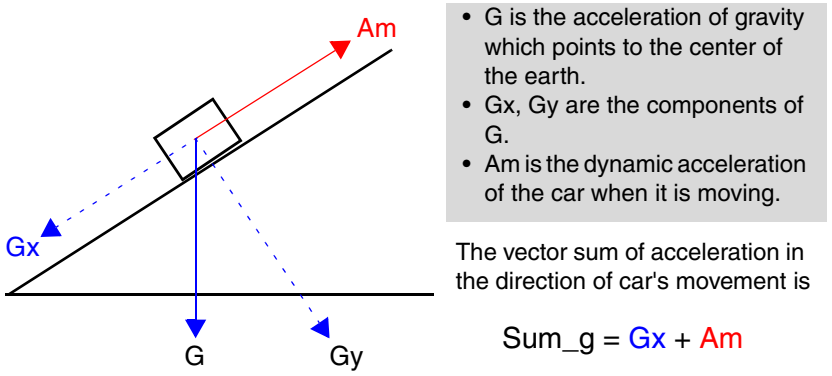

Figure 1. Acceleration components

To detect a car's movement or start (from a stop), we must set a trigger threshold limit (TRANSIENT_THS Register, 0x1F) of transient detection based on the value of Sum_g. This is difficult because we are not sure of the value of Gx, since Gx is a static component and will vary with the change of slope. However,

if we use a high-pass filter, then the static component Gx can be filtered out; therefore, the threshold is only determined by Am.

Table 1, “Car motion and acceleration values” shows real Am data when the module was installed on cars.

Table 1. Car motion and acceleration values

Car is moving				Car is not moving (stopped)		
x(g)	y(g)	z(g)		x(g)	y(g)	z(g)
+0.0107	+0.1152	+0.9951		+0.0224	+0.0078	+1.0214
+0.0156	+0.0009	+0.9990		+0.0175	-0.0410	+1.0009
+0.1171	+0.0625	+0.9677		+0.0117	-0.0175	+1.0185
+0.0878	+0.0683	+1.0361		+0.0205	+0.0117	+1.0195
+0.1210	+0.0732	+1.0126		+0.0214	-0.0341	+1.0214
+0.0449	+0.0478	+1.0576		+0.0126	-0.0214	+1.0312
+0.1435	+0.0332	+1.0048		+0.0195	+0.0136	+1.0195
+0.0771	+0.0957	+0.9326		+0.0224	-0.0283	+1.0078
+0.1591	+0.0263	+0.9521		+0.0136	-0.0341	+1.0253
+0.1015	+0.0478	+0.9326		+0.0214	+0.0253	+1.0302
+0.0712	+0.1025	+1.0195		+0.0136	-0.0302	+1.0009
+0.1298	+0.0527	+0.9765		+0.0195	-0.0302	+1.0117
+0.1181	+0.0195	+0.9980		+0.0136	+0.0244	+1.0175
+0.0976	+0.0810	+0.9765		+0.0175	-0.0136	+1.0068
+0.1171	+0.0478	+1.0058		+0.0156	-0.0371	+1.0244
+0.1376	+0.0605	+0.9443		+0.0097	-0.0029	+1.0341
+0.1005	+0.0410	+0.9472		+0.0068	+0.0205	+1.0146
+0.0166	-0.0019	+1.0488		+0.0214	-0.0322	+1.0166
+0.1005	-0.0351	+1.0117		+0.0146	+0.0214	+1.0273
+0.0107	-0.0449	+1.0751		+0.0234	+0.0234	+1.0126
The MMA8452Q +x-axis points in the direction of the car's forward movement. "+" represents positive value (accelerate), "-" represents negative value (decelerate)						

From the positive output value of the x-axis in the “Car is moving” section of Table 1, we know that the car is now accelerating. The TRANSIENT_THS Register (0x1F) can be set according to the data we captured. Here we set the TRANSIENT_THS register to 0.126 g. To guarantee detection on the other two axes (Y, Z), we enable the Y and Z axis. Any movement on any axis (X or Y or Z) exceeding 0.126 g will be detected by the accelerometer.

```
// transient threshold=0.126g
EI2C1_SendChar(MMA8452_ADDR,REG52_TRANSIENT_THS,0x01);
```

2.3 Interrupt connections

Figure 2 shows the connection of sensor's INT pin and MCU.

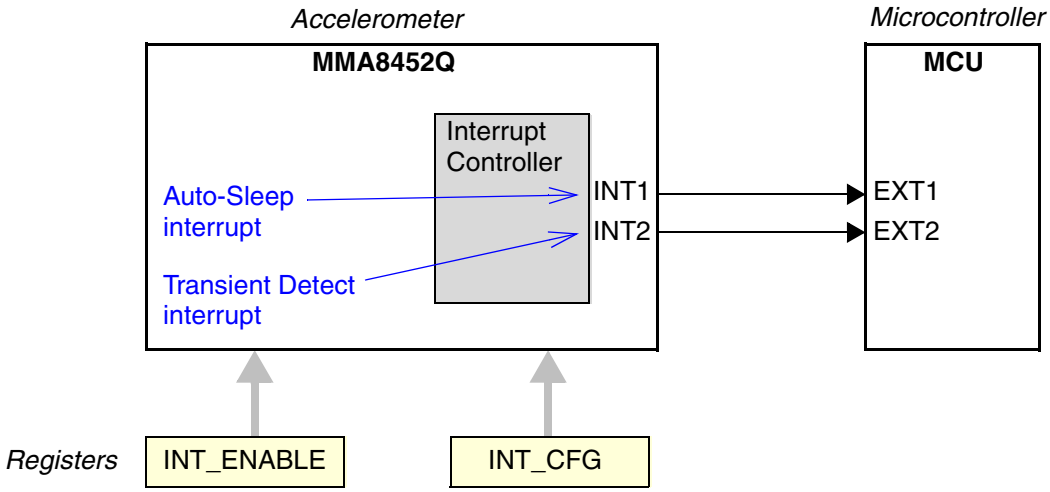

Figure 2. Interrupt connections

The MMA8452Q has two interrupt pins (INT1, INT2), and any interrupt source could be routed to either of them. INT1 and INT2 are connected to the external interrupt pins of the MCU. Our design uses two interrupt sources: transient and auto-wake/sleep, which are routed to different INT pins:

- Transient interrupt is routed to INT2
- Auto-wake/sleep is routed to INT1

Why route to different INT pins?

The transient interrupt can be auto-cleared by setting the ELE bit (Transient_CFG Register, 0x1D) to 0, while the auto-wake/sleep interrupt only can be cleared by reading the SYSMOD Register (0x0B). If these two interrupts route to the same INT pin, then the MCU cannot distinguish which interrupt comes (is it an auto-wake/sleep or transient interrupt?). Thus, the MCU will have to respond to the ISR every time, just to clear the interrupt. This is not what we expect, because the transient interrupt will always be generated when the car is moving.

The purpose of using two interrupt pins is so that the MCU can avoid clearing the sensor's interrupt frequently (just responding to the auto-wake/sleep interrupt), thus saving system power. In fact, connecting the INT2 pin (of MMA8452) to the EXT2 pin (of MCU) is not required; we can leave INT2 unconnected (floating), however the internal transient interrupt source must still be routed to INT2.

Can we disable transient interrupts because the INT2 pin connection seems useless to us? No, because we use transient detection as a wakeup source for the auto-wake/sleep function. We have to enable transient interrupt or the accelerometer will not be able to wake up from sleep mode. The corresponding register setting is:

```
// normal mode, auto-sleep enabled
EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG2,0x04);
```

```
// transient enabled wake up from auto-sleep
EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG3,0x40);

// ASLP INT enabled, transient INT enabled
EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG4,0xA0);

// auto-sleep -> INT1, transient -> INT2
EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG5,0x80);
```

Besides the points mentioned previously, there are other registers that you have to configure, such as dynamic range, output data rate (ODR), and others. See Section 3, “Demo Code”.

3 Demo Code

Table 2. Demo functions

Function	Description
`void mma845x_init(void)`	Initializes the MMA8452Q device.
`ISR(ext_int)`	Interrupt routine The ISR routine is mainly used to clear the interrupts, but you can use ISR to determine the car's current status (moving or stop) for debugging purposes.

Example 1. Demo

```
void mma845x_init(void)
{
   // rst,normal mode
   EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG2,0x40);
   // ODR=50,sleep ODR=12.5,enter standby mode
   EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG1,0x64);
   // normal mode, auto-sleep enabled
   EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG2,0x04);
   // transient enabled wake up from auto-sleep
   EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG3,0x40);
   // ASLP INT enabled, Transient INT enabled
   EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG4,0xA0);
   // auto-sleep -> INT1,transient -> INT2
   EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG5,0x80);
   // HPF output enabled , 2g mode
   EI2C1_SendChar(MMA8452_ADDR,REG52_XYZ_DATA_CFG,0x10);
   // HPF=2Hz
   EI2C1_SendChar(MMA8452_ADDR,REG52_HP_FILTER_CUTOFF,0x00);

   // P/L disabled
   EI2C1_SendChar(MMA8452_ADDR,REG52_PL_CFG,0x80);

   // MT/FF disabled
   EI2C1_SendChar(MMA8452_ADDR,REG52_FF_MT_CFG,0x00);

   // TRANSIENT
   // ELE latch disabled, HPF enable, transient x/y/z axes enabled
```

```
   EI2C1_SendChar(MMA8452_ADDR,REG52_TRANSIENT_CFG,0x0E);
   // transient threshold=0.126g
   EI2C1_SendChar(MMA8452_ADDR,REG52_TRANSIENT_THS,0x01);
   EI2C1_SendChar(MMA8452_ADDR,REG52_TRANSIENT_COUNT,0x00);

   // tap/double tap disabled
   EI2C1_SendChar(MMA8452_ADDR,REG52_PULSE_CFG,0x00);
   // the minimum time required to be judged from move to stop,set to 3 sec
   // the time can also be set up to 81 sec (when set to 0xFF)
   EI2C1_SendChar(MMA8452_ADDR,REG52_ASLP_COUNT,0x0A);
   EI2C1_SendChar(MMA8452_ADDR,REG52_CTRL_REG1,0x65);
}
ISR(ext_int)
{
  u8 cnt;
  KBISC_KBACK = 0x01;
  EI2C1_RecvBlock(MMA8452_ADDR,REG52_INT_SOURCE,rcv_buf,1,&cnt);
//EI2C1_RecvBlock(MMA8452_ADDR,REG52_TRANSIENT_SRC,rcv_buf+1,1,&cnt);
  EI2C1_RecvBlock(MMA8452_ADDR,REG52_SYSMOD,rcv_buf+2,1,&cnt);
 // wake
if(((rcv_buf[0]&0x80)==0x80)&&((rcv_buf[0]&0x20)==0x20))       {
        AS1_SendStr("car move\r\n",10);
    }
// sleep
else if(((rcv_buf[0]&0x80)==0x80)&&((rcv_buf[0]&0x20)==0x00))       {
        AS1_SendStr("car stop\r\n",10);
    }
}
```

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4601
Rev 0
12/2012