

## 控制三轴传感器 (DMARD08)范例说明

Example Brief for 32-bit NuMicro® Family

Rev. 1.00 — Sep. 18, 2013

### 文档信息

概 述	使用I2C控制三轴传感器 (DMARD08) 模块
适用范围	M051系列

## 目 录

---

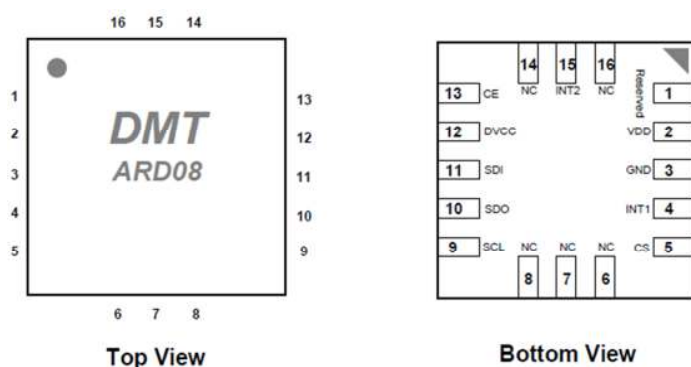
<b>1</b>	<b>简述.....</b>	<b>3</b>
1.1	环境需求与设定.....	3
1.2	芯片设定.....	3
<b>2</b>	<b>程序介绍 .....</b>	<b>4</b>
2.1	Main函数.....	4
2.2	ARD08_SingleWrite函数 .....	5
2.3	ARD08_SingleRead函数 .....	6
2.4	ARD08_MultipleRead函数.....	7
<b>3</b>	<b>执行结果 .....</b>	<b>10</b>

## 1 简述

使用 I2C 控制三轴传感器 (DMARD08)模块, 并且通过 keil 串口调试窗口打印 X, Y, Z 三轴与温度的输出值.

### 1.1 环境需求与设定

- 需搭配 M051SeriesBSP\_CMSIS\_v2.01.002 (可至新唐公司网站下载, [http://www.nuvoton.com/NuvotonMOSS/Community/ProductInfo.aspx?tp\\_GUID=4b47b09d-b116-4ccd-aa85-31e261a87d30](http://www.nuvoton.com/NuvotonMOSS/Community/ProductInfo.aspx?tp_GUID=4b47b09d-b116-4ccd-aa85-31e261a87d30)), 将此范例放至 `\SampleCode\Driver` 目录下.
- 需要使能 SEMIHOST 功能, 经由 KEIL 串行调试窗口(Uart 1)打印讯息
- 选用 DMARD08 三轴传感器模块
- 三轴传感器提供 I2C 与 SPI 两种接口, 这里是使用 I2C 接口.
- 脚位连接:
  - P2.0 → DMT ARD08 Pin 13 (CE)
  - P3.4 → DMT ARD08 Pin 9 (SDI)
  - P3.5 → DMT ARD08 Pin 11 (SCL)



- I2C Bus 需要加上拉电阻. DMARD08 的建议值为 4.7k ohm.

### 1.2 芯片设定

- HCLK = PLL (48 MHz)
- P2.0 配置为 GPIO 输出模式
- P3.4 – 5 分别配置为 I2C SDA 与 SCL
- I2C 设定
  - I2C Bus clock = 200 kHz
  - 操作在主模式 (通过发送一个 START 讯号)

## 2 程序介绍

### 2.1 Main 函数

对系统时钟, GPIO与I2C0做初始化, 将P2.0配置成输出模式, P3.4 – 5 配置为I2C模式, I2C Bus clock设定为200 kHz.

```
int main (void)
{
    ...
    /* 初始化系统,系统时钟和GPIO配置 */
    SYS_Init();
    ...
    /* 配置P2.0为输出模式 */
    _GPIO_SET_PIN_MODE(P2, 0, GPIO_PMD_OUTPUT);

    /* 设置P2.0输出低电平,传感器操作在待机模式 */
    P20 = 0;

    /* 初始化I2C0,I2C0总线时钟 = 200 kHz */
    I2C0_Init();
    ...
}
```

为了使能DMARD08三轴传感器, 需要将DMARD08 Pin 13 (CE)设为高电平. DMARD08三轴传感器的I2C slave address的LSB会依照脚位SDO的电平高低而有所不同, 若SDO的电平为low, 则slave address = 0x1C; 若SDO为high, 则slave address = 0x1D. 此示例SDO接至GND, 所以slave address为0x1C. 之后再对DMARD08三轴传感器模块做初始化设定, 并且将传感器4个控制寄存器读回确认.

```
/* 设置P2.0为高电平,传感器操作在工作模式 */
P20 = 1;

/* G-Sensor Pin SDO is connected to GND. Thus, G-Sensor slave address=0x1C */
/* 传感器的SDO脚是接地的,所以传感器作为从设备的地址为0x1C */
g_u8DeviceAddr = 0x1C;

/* 写ARD08寄存器 */
ARD08_Singlewrite(0x08, 0x03);           // 采样周期 = 2.925 ms, 采样频率 = 342 Hz
                                           // 移动平均长度 = 1, 带宽 = No Filter

/* 读ARD08控制寄存器 */
u8RegData = ARD08_SingleRead(0x08);
```

```
printf("Control Register (0x08): 0x%X\r\n", u8RegData);
u8RegData = ARD08_SingleRead(0x09);
printf("Control Register (0x09): 0x%X\r\n", u8RegData);
u8RegData = ARD08_SingleRead(0x0A);
printf("Control Register (0x0A): 0x%X\r\n", u8RegData);
u8RegData = ARD08_SingleRead(0x0B);
printf("Control Register (0x0B): 0x%X\r\n\r\n", u8RegData);
```

分别读出三轴传感器模块X, Y, Z三轴与温度转换后的11-bit数据. 若读出11-bit数据的MSB为1, 代表数据为负值. X, Y, Z三轴的灵敏度为256 LSB/g, 而输出值0x00代表加速度为0 g. 温度的灵敏度为16 LSB/°C, 输出值0x00代表温度为25°C.

```
i16Tout = ARD08_MultipleRead(ARD08_REG_TOUT); // 读温度值
i16Xout = ARD08_MultipleRead(ARD08_REG_XOUT); // 读X轴值
i16Yout = ARD08_MultipleRead(ARD08_REG_YOUT); // 读Y轴值
i16Zout = ARD08_MultipleRead(ARD08_REG_ZOUT); // 读Z轴值

/* 如果输出值最高位为1, 则表示输出值为负值 */
i16Tout_d = i16Tout | ((i16Tout & 0x400)? 0xF800:0);
i16Xout_d = i16Xout | ((i16Xout & 0x400)? 0xF800:0);
i16Yout_d = i16Yout | ((i16Yout & 0x400)? 0xF800:0);
i16Zout_d = i16Zout | ((i16Zout & 0x400)? 0xF800:0);

/* 串口打印三轴输出值以及相应的加速度值 */
/* 灵敏度为256 LSB/g, 中间值(0x00)表示 0 g */
printf("X Output: %.4f g (0x%X)\r\n", (float)i16Xout_d / (float)256, i16Xout);
printf("Y Output: %.4f g (0x%X)\r\n", (float)i16Yout_d / (float)256, i16Yout);
printf("Z Output: %.4f g (0x%X)\r\n", (float)i16Zout_d / (float)256, i16Zout);

/* 串口打印温度寄存器值和实际温度值 */
/* 1度 = 16 LSB, 中间值(0x00)表示 25 摄氏度 */
printf("Temperature: %.2f degrees C (0x%X)\r\n\r\n", ((float)i16Tout_d / (float)16)+25, i16Tout);
```

## 2.2 ARD08\_SingleWrite 函数

ARD08\_SingleWrite函数则是依照三轴传感器的Single Write传输协议, 先写入寄存器地址, 再将数据写入.

Single Write

ST	Slave Address + RW	A	Reg Address	A	Data	A	SP
----	--------------------	---	-------------	---	------	---	----

```
void ARD08_Singlewrite(uint8_t u8Reg, uint8_t u8Data)
```

```
{
    /* 发送起始位 */
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STA);
    _I2C_WAIT_READY(I2C0);

    /* 写传感器从设备地址+写位 */
    I2C0->I2CDAT = g_u8DeviceAddr << 1;
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
    _I2C_WAIT_READY(I2C0);

    /* 写传感器寄存器地址 */
    I2C0->I2CDAT = u8Reg;
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
    _I2C_WAIT_READY(I2C0);

    /* 写数据到传感器寄存器 */
    I2C0->I2CDAT = u8Data;
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
    _I2C_WAIT_READY(I2C0);

    /* 发送停止位 */
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STO_SI);

    SYS_SysTickDelay(10);
}
```

## 2.3 ARD08\_SingleRead 函数

ARD08\_SingleRead函数则是依照三轴传感器的Single Read传输协议, 将寄存器的数据读出.

Single Read

ST	Slave Address + RW	A	Reg Address	A	SR	Slave Address + RW	A
Data	NA	SP					

```
uint8_t ARD08_SingleRead(uint8_t u8Reg)
{
    uint8_t u8data;

    /* 发送起始位 */
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STA);
    _I2C_WAIT_READY(I2C0);
```

```
/* 写传感器从设备地址+写位 */
I2C0->I2CDAT = g_u8DeviceAddr << 1;
_I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
_I2C_WAIT_READY(I2C0);

/* 写传感器寄存器地址 */
I2C0->I2CDAT = u8Reg;
_I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
_I2C_WAIT_READY(I2C0);

/* 发送重复起始位 */
_I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STA_SI);
_I2C_WAIT_READY(I2C0);

/* 写传感器从设备地址+读位 */
I2C0->I2CDAT = ((g_u8DeviceAddr << 1) | 0x01);
_I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
_I2C_WAIT_READY(I2C0);

/* 读传感器寄存器数据并返回NACK */
_I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
_I2C_WAIT_READY(I2C0);

u8data = I2C0->I2CDAT;

/* 发送停止位 */
_I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STO_SI);

SYS_SysTickDelay(10);

return u8data;
}
```

## 2.4 ARD08\_MultipleRead 函数

ARD08\_MultipleRead函数则是依照三轴传感器的Multiple Read传输协议, 用来读取X, Y, Z三轴与温度的输出寄存器后, 再将其转换成11-bit的数据.

### Multiple Read

ST	Slave Address + RW		A	Reg Address		A	SR	Slave Address + RW		A
Data		A	Data		NA	SP				

```
uint16_t ARD08_MultipleRead(uint8_t u8Reg)
{
    uint8_t u8highByte, u8lowByte;

    /*发送起始位 */
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STA);
    _I2C_WAIT_READY(I2C0);

    /* 写传感器从设备地址+写位 */
    I2C0->I2CDAT = g_u8DeviceAddr << 1;
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
    _I2C_WAIT_READY(I2C0);

    /* 写传感器寄存器地址 */
    I2C0->I2CDAT = u8Reg;
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
    _I2C_WAIT_READY(I2C0);

    /* 发送重复起始位 */
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STA_SI);
    _I2C_WAIT_READY(I2C0);

    /* 写传感器从设备地址+读位 */
    I2C0->I2CDAT = ((g_u8DeviceAddr << 1) | 0x01);
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
    _I2C_WAIT_READY(I2C0);

    /* 读传感器寄存器数据并返回ACK */
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI_AA);
    _I2C_WAIT_READY(I2C0);

    u8lowByte = I2C0->I2CDAT;

    /* 读传感器寄存器数据并返回NACK */
    _I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_SI);
    _I2C_WAIT_READY(I2C0);
}
```



```
u8highByte = I2C0->I2CDAT;

/* 发送停止位 */
_I2C_SET_CONTROL_BITS(I2C0, I2C_I2CON_STO_SI);

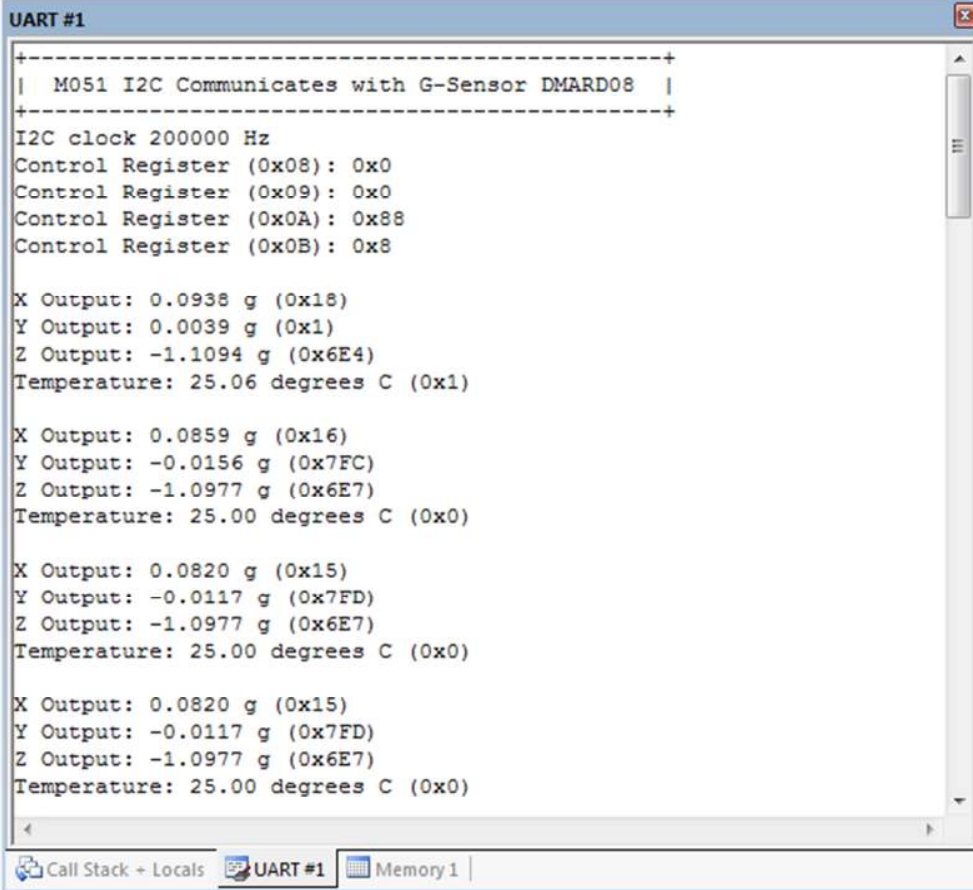
SYS_SysTickDelay(10);

return ((u8lowByte << 3) | (u8highByte & 0x7));
}
```

### 3 执行结果

开启 KEIL Serial Windows, 由打印出的讯息, 可得知三轴传感器输出的 X, Y, Z 三轴与温度 11-bit 原始数据, 以及转换后的加速度与温度值.

显示结果如下:



```
UART #1
+-----+
| M051 I2C Communicates with G-Sensor DMARD08 |
+-----+
I2C clock 200000 Hz
Control Register (0x08): 0x0
Control Register (0x09): 0x0
Control Register (0x0A): 0x88
Control Register (0x0B): 0x8

X Output: 0.0938 g (0x18)
Y Output: 0.0039 g (0x1)
Z Output: -1.1094 g (0x6E4)
Temperature: 25.06 degrees C (0x1)

X Output: 0.0859 g (0x16)
Y Output: -0.0156 g (0x7FC)
Z Output: -1.0977 g (0x6E7)
Temperature: 25.00 degrees C (0x0)

X Output: 0.0820 g (0x15)
Y Output: -0.0117 g (0x7FD)
Z Output: -1.0977 g (0x6E7)
Temperature: 25.00 degrees C (0x0)

X Output: 0.0820 g (0x15)
Y Output: -0.0117 g (0x7FD)
Z Output: -1.0977 g (0x6E7)
Temperature: 25.00 degrees C (0x0)
```

### 版本历史

Revision	Date	Description
1.00	Sep. 18, 2013	初始发布.

### **Important Notice**

---

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur. Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

---

Please note that all data and specifications are subject to change without notice. All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.