How to run the project: build → Run main.c file
Main.c file is under 704P2_Compendium > COMPSYS704 > Projects > STM32L476JG-SensorTile > Applications > ALLMEMS1 > Src

Fixed define values are under /*Defined values*/

```
97
98  /* Defined values ------------------------------------------------------------ */
99  #define PI 3.141592
100 #define NUM_SAMPLES 5
101 #define STEP_START_THRESHOLD 1100
102 #define STEP_FINISH_THRESHOLD 900
103 #define MAGNITUDE_THRESHOLD 0
104 #define STRIDE_LENGTH 57
105
```

Public variables are under /*Defined Variables*/

```
181
182 /* Defined Variables ---------------------------------------------------------*/
183
184 static int stepCount =0;            // check for static and just int
185 bool stepDetected = false;
186 bool stepCounted = false;
187 float oldAccZ =900;
188 //static float currentHeading =0;
189 static float initialHeading = -1;
190 static float totalDistance =0.0;
191 static float currentPositionX = 0.0;
192 static float currentPositionY = 0.0;
193
```

Address for Accelerometer and magnetometer are under /*for ACC*/ and /*for MAG*/

```
60 /* Private define ------------------------------------------------------------*/
61
62 /* for ACC */
63 #define STATUS_REG_A 0x27
64
65 #define CTRL_REG1_A 0x20
66 #define CTRL_REG2_A 0x21
67 #define CTRL_REG3_A 0x22
68 #define CTRL_REG4_A 0x23
69
70 #define OUT_X_L_A 0x28
71 #define OUT_X_H_A 0x29
72
73 #define OUT_Y_L_A 0x2A
74 #define OUT_Y_H_A 0x2B
75
76 #define OUT_Z_L_A 0x2C
77 #define OUT_Z_H_A 0x2D
78
79
80 /* for MAG */
81 #define ZYXDA_BIT 0x08
82 #define STATUS_REG_M  0x67
83
84 #define CFG_REG_A_M   0x60
85 #define CFG_REG_B_M   0x61
86 #define CFG_REG_C_M   0x62
87
88
89 #define OUTX_L_REG_M  0x68
90 #define OUTX_H_REG_M  0x69
91
92 #define OUTY_L_REG_M  0x6A
93 #define OUTY_H_REG_M  0x6B
94
95 #define OUTZ_L_REG_M  0x6C
96 #define OUTZ_H_REG_M  0x6D
97
```

startMag() : void
startAcc() : void
readMag() : void
readAcc() : void

There are four functions and *main(void)*.

The magnetometer is initialised in *startMag()* function and the accelerometer is initialised in *startAcc()* function.

```
253 static void startMag() {
254     //#CS704 - Write SPI commands to initiliase Magnetometer
255     uint8_t inData[10]; // Buffer for data to be written to the registers
256     // Set specific operational mode and enable temperature compensation


266
267 static void startAcc() {
268     //#CS704 - Write SPI commands to initiliase Accelerometer
269     uint8_t inData[10];
270     // Reset all the settings in control register 2
```

The data values are read and converted to appropriate units in *readMag()* function and *readAcc()* function. Also they are sampled and averaged. (magnetometer with hard iron compensation as well).

```
284 static void readMag() {
285     // Function to read data from the Magnetometer
286     // The function reads the x, y, and z magnetic fields, averages the readings, and applies a hard iron correction.
287
288     uint8_t magStatus;      // checking for magnetometer status
289     uint8_t magLSB, magMSB; // variables to hold LSB and MSB
290     int16_t OUTX_NOST_M, OUTY_NOST_M, OUTZ_NOST_M;  // variables to hold combined LSB and MSB

351 static void readAcc() {
352     uint8_t accStatus;
353     uint8_t accLSB, accMSB;
354     int16_t OUTX_NOST_A, OUTY_NOST_A, OUTZ_NOST_A;
355
356     const float accFullScale = 2.0;
```

In main(void), heading calculation, step detection, and position calculation are performed.

Heading calculation is under /* Calculate Heading */

```
/* Calculate Heading */
float magX_h = MAG_Value.x; // in milli Gauss
float magY_h = MAG_Value.y;
float magZ_h = MAG_Value.z;

//arctang calcuation and radians to degrees
float currentHeading = (atan2(magY_h,magX_h)*180)/PI;
```

Step detection is under /* Step detection */

```
/* Step detection */

float accX_mg = ACC_Value.x;     // in milli-g
float accY_mg = ACC_Value.y;
float accZ_mg = ACC_Value.z;

int16_t accX_int = (int16_t)accX_mg;
```

Position calculation is under /* Position calculation*/

```
559     /* Position calculation */
560
561     // Calculate the change in position after a step has been counted
562     if (stepCounted){
563         float stepDistance = STRIDE_LENGTH; // one step distance
564         totalDistance += stepDistance; // total distance
565
566         float positionChangeX= stepDistance * sin(relativeHeading* PI/180.0);
567         float positionChangeY= stepDistance * cos(relativeHeading* PI/180.0);
```

Values stored in COMP_Value which will be sent to the application via bluetooth

```
579     // Store the current position and relative heading in the COMP_Value
580     COMP_Value.x = (int16_t)currentPositionX;
581     COMP_Value.y = (int16_t)currentPositionY;
582     COMP_Value.Heading = relativeHeadingInt *10;
583
584     }
```