Magnetic Field Sensors

A common magnetic field sensor is a Hall effect sensor, which is a transducer with output voltage changes with the strength of the magnetic field at the sensor. Hall sensors have been used as speed sensor, position sensor, rotation sensor, and other applications. The particular Hall effect sensor chosen for this section is Panasonic’s DN6848SE. The datasheet states that “the sensor consists of an amplifier circuit, a Schmidt circuit, a stabilized power supply, and a temperature compensation circuit all integrated on a single IC. The Hall element output is amplified by the amplifier circuit, and converted into the corresponding digital signals through the Schmidt circuit that TTL and MOS IC are directly drivable.” Pictures of the DN6848SE and web link of the datasheet are shown below.

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Part Number</th>
<th>Weblink for the part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panasonic</td>
<td>DN6848SE</td>
<td><a href="http://www.datasheetcatalog.org/datasheet/panasonic/SPC00003CEB.pdf">http://www.datasheetcatalog.org/datasheet/panasonic/SPC00003CEB.pdf</a></td>
<td>Hall effect sensor</td>
</tr>
</tbody>
</table>
Hardware interfacing to the Freescale 9S12C32 MCU

There are three pins in the DN6848SE package. The power supply (4.5 to 16 V) is connected to pin 1. Ground is connected to pin 2. Pin 3 is the output signal and is pulled high. A schematic diagram is shown below. At the absence of magnetic field, pin 3 output voltage is the power supply voltage Vcc. At the presence of magnetic field such as when a magnet is directly at the sensor, pin 3 drops to ground level. Pin 3 can be connected straight to the MCU or through a 1 kohm resistor. In this case, Vcc should be 5 V. A schematic diagram is shown below.

If the sensor is used for measuring rotating speed, one or more magnets are to be attached to the rotating object. A stationary Hall sensor is place nearby the object. When a magnet is directly over the Hall sensor, it induces logic low signal at pin 3. Since pin 3 is normally high, a sequence of pulses is generated at pin 3 when the magnets are passing by the Hall sensor. This sequence of pulses can be fed into the IRQ pin of the MCU. The IRQ pin can be initialized with negative edge triggering. The IRQ interrupt service routine will increase a counter for every falling edge at the IRQ. The number of pulses registered over a predetermined period of time can be used to compute the speed of rotation. The code snippets below cover this speed measurement application.

Software development

The following code snippet initializes the IRQ for falling edge triggering and enables it.

```c
void init_IRQ(void) {
    EnableInterrupts;  // clear the I bit allowing interrupts
    INTCR_IRQE = 1;    // choose falling edge triggering
    INTCR_IRQEN = 1;   //enable IRQ interrupt
}
```

The IRQ interrupt service routine is shown below. It increases a counter “count” by one for every falling edge occurring at the IRQ pin.

```c
void interrupt 6 count_pulses(void) {  //the #6 specifies IRQ type interrupt
    count++;  // count is a global, integer variable, initialized with 0,
    return;
}
```
To compute the speed of rotation, S, in RPM, we need to make some assumptions. First, assume that there are N magnets equally spaced around the circumference of the rotating object. Such will produce N pulses per revolution. Second, suppose that the pulses are counted for T seconds. Then at the end of the T seconds, the variable count is given by the equation

$$\text{count} = \frac{S}{60} \times N \times T$$

Therefore the speed of rotation in RPM can be computed by the equation

$$S = \frac{60 \times \text{count}}{N \times T}$$

To calculate S, the code snippet below can be called. In the code, T is arbitrarily chosen to be 2 sec. and assume N=3. The total number of pulses counted in these two seconds will be stored in the global variable count, which is initialized to be zero before counting. T can be chosen as some other value. If the RPM is fast, T can be chosen to be smaller. If the RPM is slow, T should be sufficient large so that there is enough number of pulses accumulated for accurate calculations.

```c
void compute_RPM(void){
    int N=3;
    int T=2000;        // T is equal to 2000 msec
    count=0;          // count is a global, integer variable
    waitms(T);        // delay T msec., 2 second delay for counting pulses
                      // waitms() was covered in class
    S=60*count/(N*T);  //S is a global, float variable
    return;
}
```