NEC

Application Note

MR SENSOR Solution Guide

Examples of MR sensor used in industrial applications

Target Devices:
- MR sensors
- Microcontrollers (78K0, 78K0R, V850)

Document No. S19854EE1V0AN00
Date published July 2009

Printed in Germany

Legal Notes

- **The information in this document is current as of July, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**

- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.

- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

- The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.
  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime

systems, safety equipment and medical equipment (not specifically designed for life support).
"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives anddistributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and otherlegal issues may also vary from country to country.

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668, Japan
Tel: 044 4355111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554,
U.S.A.
Tel: 408 5886000
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211 65030
http://www.eu.necel.com/

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908 691133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01 30675800

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 6387200

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02 667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven,
The Netherlands
Tel: 040 2654010

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27
ZhiChunLu Haidian District,
Beijing 100083, P.R.China
Tel: 010 82351155
http://www.cn.necel.com/

NEC Electronics Shanghai Ltd.
Room 2511-2512, Bank of China
Tower,
200 Yincheng Road Central,
Pudong New Area,
Shanghai 200120, P.R. China
Tel: 021 58885400
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
12/F., Cityplaza 4,
12 Taikoo Wan Road, Hong Kong
Tel: 2886 9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R.O.C.
Tel: 02 27192377

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253 8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku, Seoul,
135-080, Korea Tel: 02-558-3737
http://www.kr.necel.com/

Table of Contents

Chapter 1 Intoduction

As part as NEC Electronics wide portfolio of electronics components, a family of devices called MR sensor can be found. These components can be used in multiple and various applications in different markets like industrial or consumer. The basics of MR sensor features and operation will be presented, before giving details of two typical applications.

Chapter 2 MR Sensor Theoretical Overview

2.1 Characteristics

NEC Electronics MR sensor is a sensor device utilizing magneto-resistance effect. It is including four magneto-resistive elements and a waveform shaping circuit in a single-chip.
MR sensor results in a magnetic sensor with high sensitivity to the magnetic field, high performance with reliability and stability.

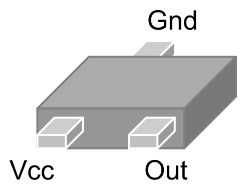

Figure 2-1 MR Sensor appearance

Equivalent block diagram of MR sensor is shown in the figure below. The MR elements are represented by R1, R2, R3 and R4. The integrated circuit with the operational amplifier outputs a signal which depends on the applied external magnetic field (and on the supplied voltage).
One of the main advantages of MR sensor is the CMOS output which directly eliminates the need of external resistor.

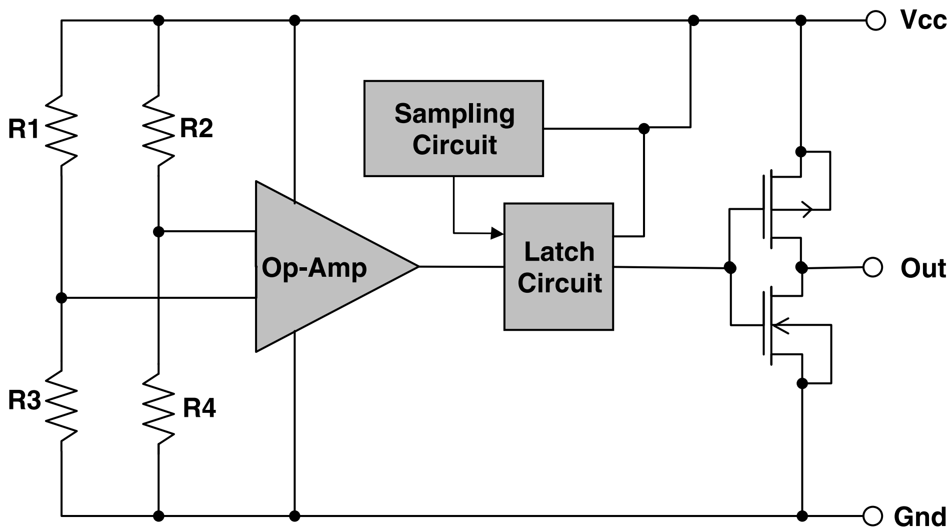

Figure 2-2 MR Sensor equivalent block diagram

The resistance of the magneto-resistive changes periodically depending on the direction of the external magnetic field and the angle of the electric current going through the MR elements.

The relation between the angle external magnetic field and electric current make [θ] and resistance value [ρ] is $\rho = \rho_0 - \Delta\rho \sin^2\theta$.

Therefore:

- the maximum resistance is when θ = 0 °
- the minimum resistance is when θ = 90°

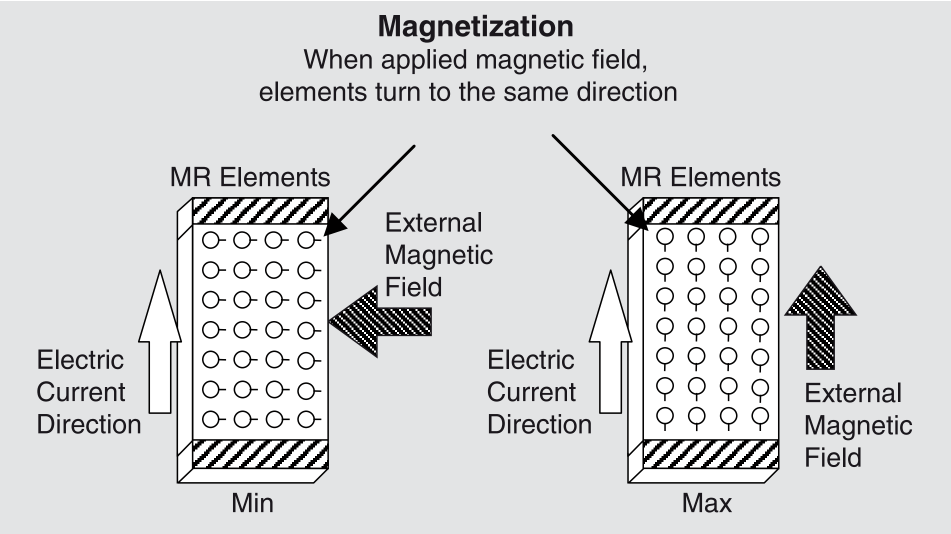

Figure 2-3 Resistance value vs Magnetic field direction

2.2 MR Sensor Operation

The four MR elements are bridged.
When external magnetic field is applied, the resistance of the MR elements change and by consequence voltage at “a” and “b” change. The integrated circuit including the comparator changes the output when offset voltage between the two points exceeds certain point.

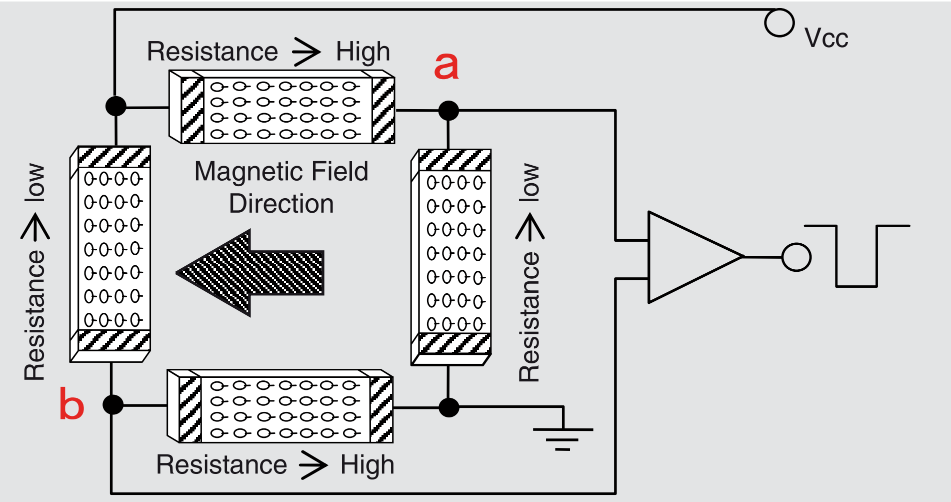

Figure 2-4 MR Sensor operation

The output voltage depends also on the supplied voltage as you can see below.

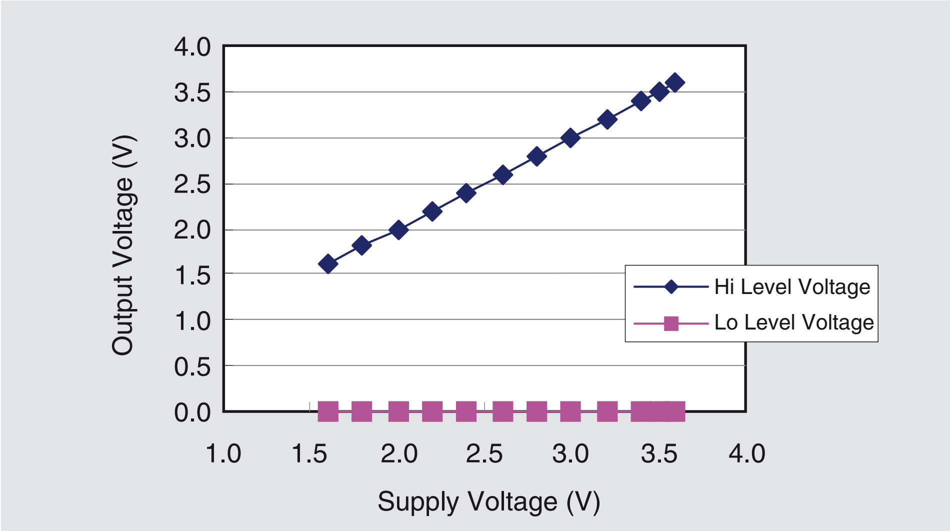

Figure 2-5 **Output level voltage VS Supply voltage**

Thus, MR sensor will detect any horizontal magnetic field:

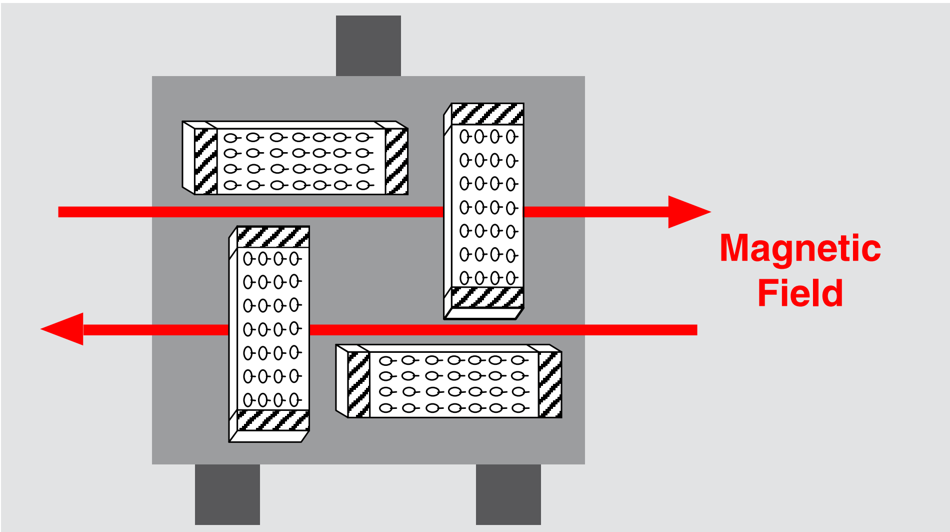

Figure 2-6 **Horizontal operating magnetic field**

MR Sensor operates with either north or south pole of sufficient strength.

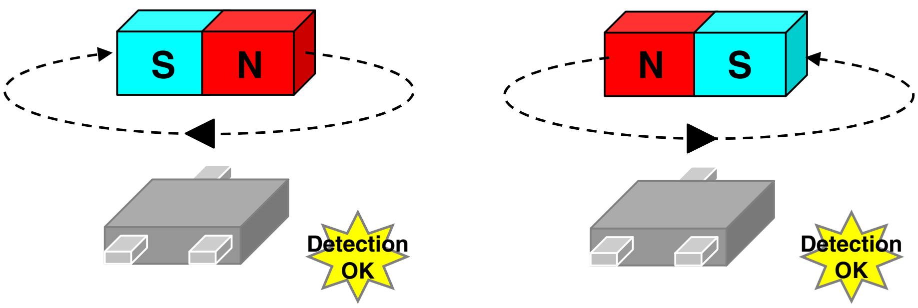

Figure 2-7 **Dual operating magnetic pole**

High sensitivity of the detection of any adequate horizontal magnetic field and the independence from the polarity of this magnetic field explain the numerous possibilities for the arrangements for the magnetic field to be detected.

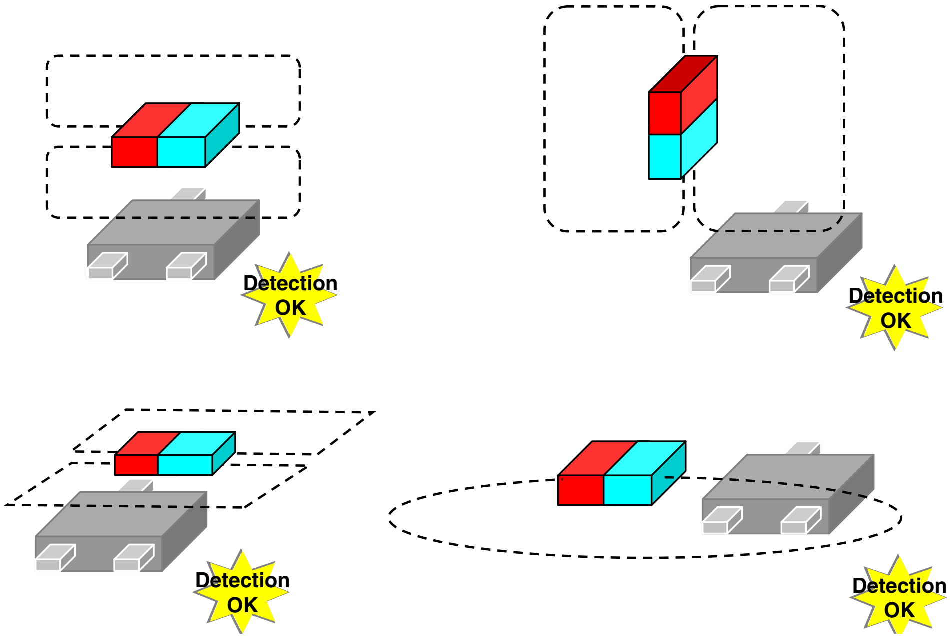

Figure 2-8 Examples of operative area

On the other side, weak or non horizontal magnetic field will not be detected.

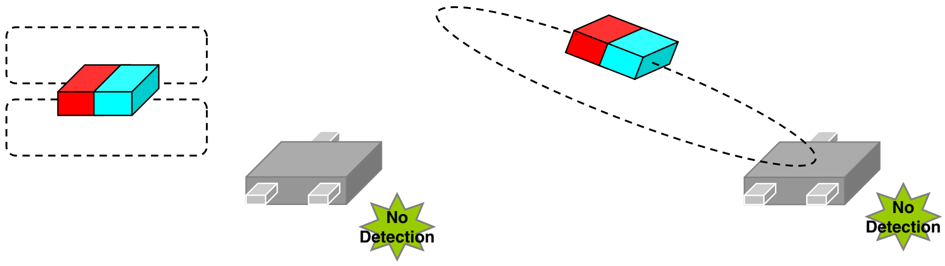

Figure 2-9 Examples of non operative area

2.3 Operational Characteristics and Operation

By default, MR sensor output is at a high level when MR sensor is power On. Then the changing of the output is defined by two levels of magnetic field force that will change the status of the output: H_{on} and H_{off}.

H_{on} is the minimum magnetic flux density that ensures that the MR sensor output is “Low”.

H_{off} is the maximum magnetic flux density that ensures that the MR sensor output is “High”.

So when a magnetic field whose flux density is below H_{off}, MR sensor is outputting High level. When a magnetic field whose flux density is higher H_{on}, MR sensor is outputting Low level.

	Magnetic flux density	Output voltage
When MR sensor is powered (no magnetic field)	H = 0 mT	High
When a magnetic field is applied	$H >= H_{on}$ (typ)	Low
When a magnetic field is applied	$H <= H_{off}$	High

2.4 Applications

Unlimited application variety as non-contact switch is possible by combining with different types of magnet, and also by arranging the positions of them. Due to its very small size and low power consumption, it is also adapted in compact/ portable system and in space limited design. NEC Electronics MR sensor is a family of more than 13 devices with different features: high speed features, low power consumption, single or dual package, 3 or 5 V operation, …

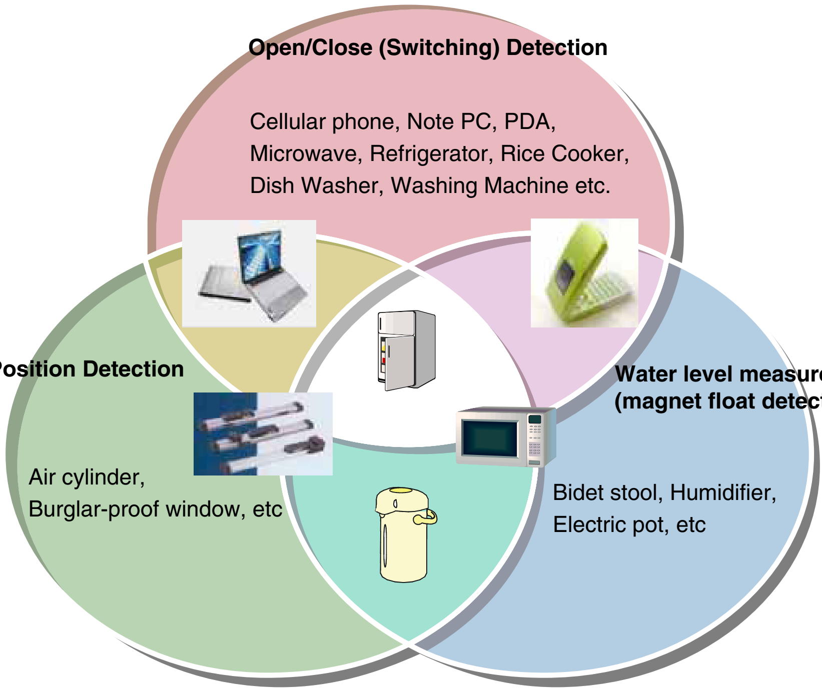

Figure 2-10 Examples of MR sensor applications

Summary of key advantages:

- Wide operating voltage range
- Wide operating ambient temperature range
- Low power consumption
- High sensitivity type of magnetic field
- Operation independent from magnetic field polarity
- RoHs compliant
- Wide range of products with different package
- Extremely compact
- Non-contact switch: high robustness

Due to its features and advantages, NEC Electronics MR Sensor can be the ideal replacement for some applications that are using Hall Sensor or Reed Switch Relay.

Compared to Hall Sensor, NEC Electronics MR Sensor with its wide operative area allows higher design flexibility.

Compared to Reed Switch relay, NEC Electronics MR Sensor is more robust and more compact, while showing lower power consumption and unlimited life time.

Associated with any microcontroller, the MR sensor became a very smart sensor for various applications. The below sections give the details of two examples of MR sensor applications used with NEC Electronics microcontrollers.

Chapter 3 MR Sensor used as Open/Close Detection

3.1 Overview

MR sensor is very easy to use as open/close position sensor.

Magnet and MR sensor will be placed in such position that:

- when they are close, MR sensor detects the magnetic field -> Close position detection
- when they are far, MR sensor does not detect the magnetic field -> Open position detection

Some applications for the MR sensor used as open/close position sensor can be mobile phone application in the consumer market, security systems in the industrial market, or big and small appliances for the detection of the door position or the lead position (fridge, oven, bread machine).

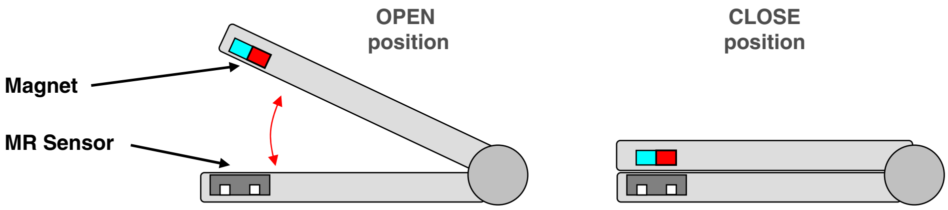

Figure 3-1 Mobile phone application

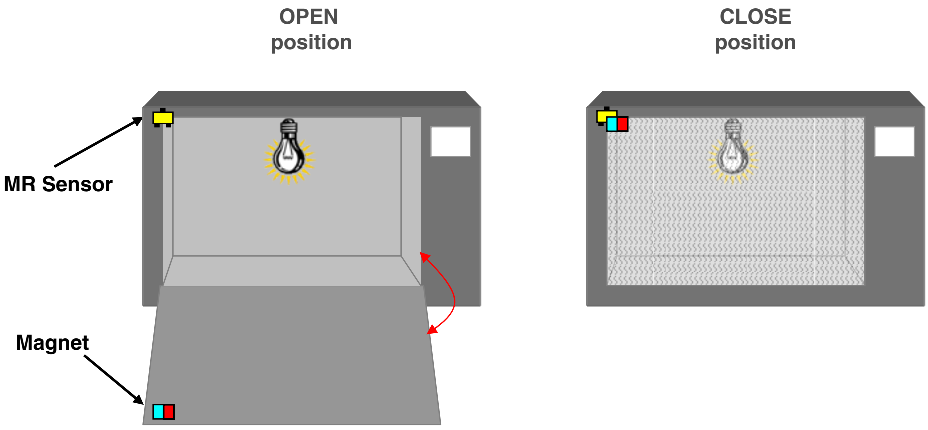

Figure 3-2 Oven application

3.2 Microcontroller Resources, Algorithm and Reference Design

In term of resources and connection to the microcontroller, they are very minimal: only one pin of the microcontroller is required to connect the MR sensor.

Depending on the application or on the requirement to detect or not real time changes:

- MR sensor can be connected to a standard I/O port set up as input. A quick check of the port status gives the position.
  - Port is LOW -> Close position
  - Port is HIGH -> Open position
- MR sensor can be connected to an external interrupt input, so in this case the edge detection gives the position.
  - Falling edge detection -> Closed detection
  - Rising edge detection -> Opened detection

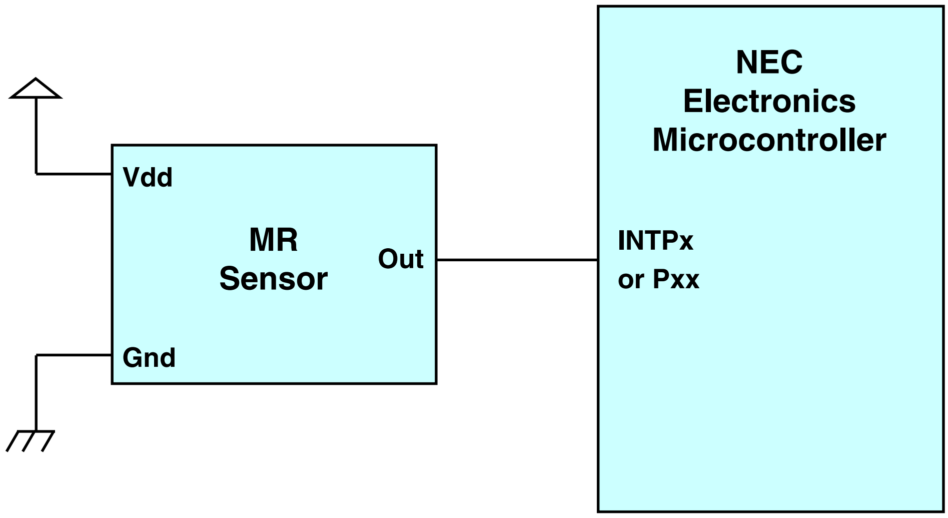

Figure 3-3 MR sensor typical connexion for open/close detection

3.3 Sample Program Example

One of the easiest applications is to change the LED status depending on the open/close position.

This sample application was developed with NEC microcontroller 78K0R/KE3-L (µPD78F1009GB). The MR sensor chosen is the MRSS23E(5) powered with the same supply voltage of the microcontroller (5V). The complete software is given as reference in "*chapter 6.1*".

1. Contents of initial settings “Low_Level_init”
   - Selecting the 20MHz external resonator as the system clock source
   - Setting I/O port P7.6: output for the LED
   - Setting external interrupt port INTP0: detection of rising and falling edge on MR sensor output

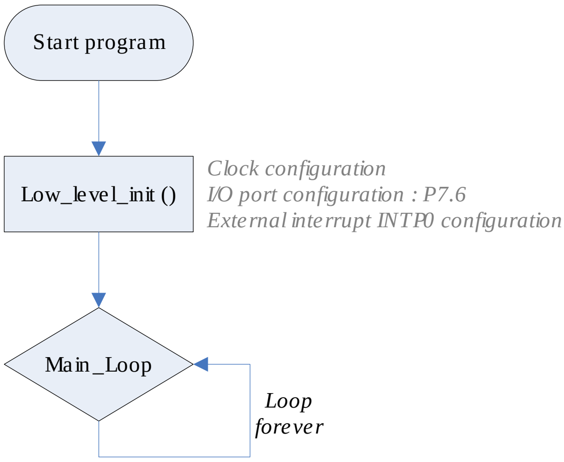

Figure 3-4 Main processing overview flowchart

2. Contents of main processing operation by interrupt "P0_ISR"
Lighting of the LED is controlled by detecting open/close position with the 78K0R/KE3-L microcontroller during the interrupt of INTP0. It is important to note that interrupt of INTP0 is generated for both edge. So to differentiate the falling and rising edge, a quick check of the port status is done.

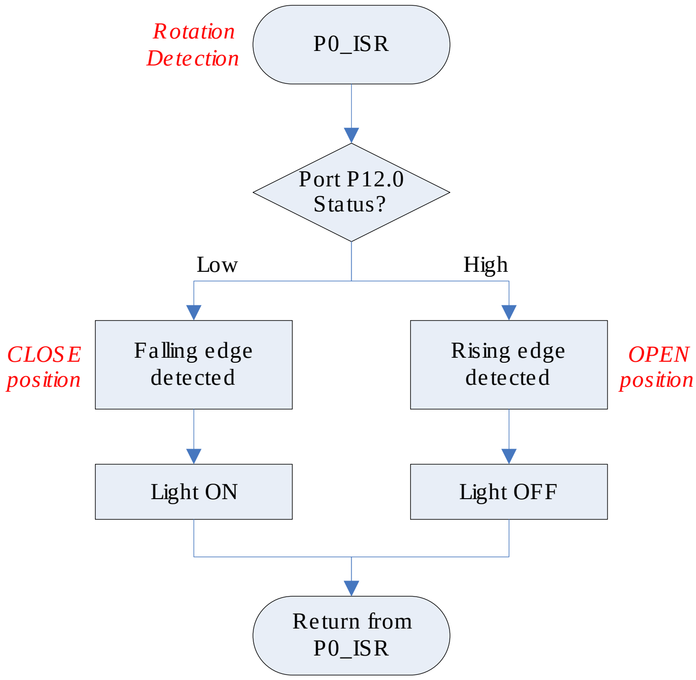

Figure 3-5 INTP0 interrupt flowchart

Chapter 4 MR Sensor used as Rotation Detection

4.1 Overview

MR sensor can be used as rotation detection for example in jog dial application (rotation wheel). Rotating selector are commonly applied on mobile phones, appliances, PDAs, computers, and others equipments for example.

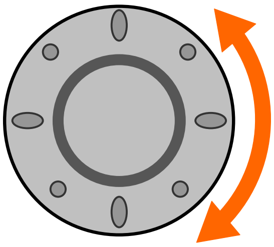

Figure 4-1 Jog dial application

Detection of the rotation:

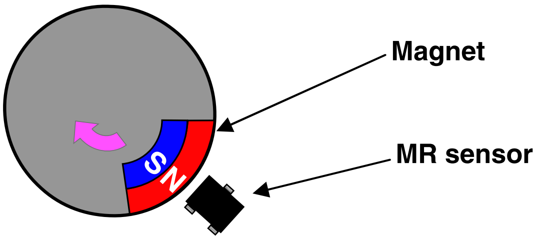

Figure 4-2 MR sensor position for rotation detection

For the rotation detection, different configuration are possible but the most popular is to have the magnet on the wheel, and the MR sensor on the static part (close to the wheel) to detect the magnetic field. Basically, by moving the rotation wheel, the MR sensor will detect the magnetic field of the magnet each time it will pass nearby.

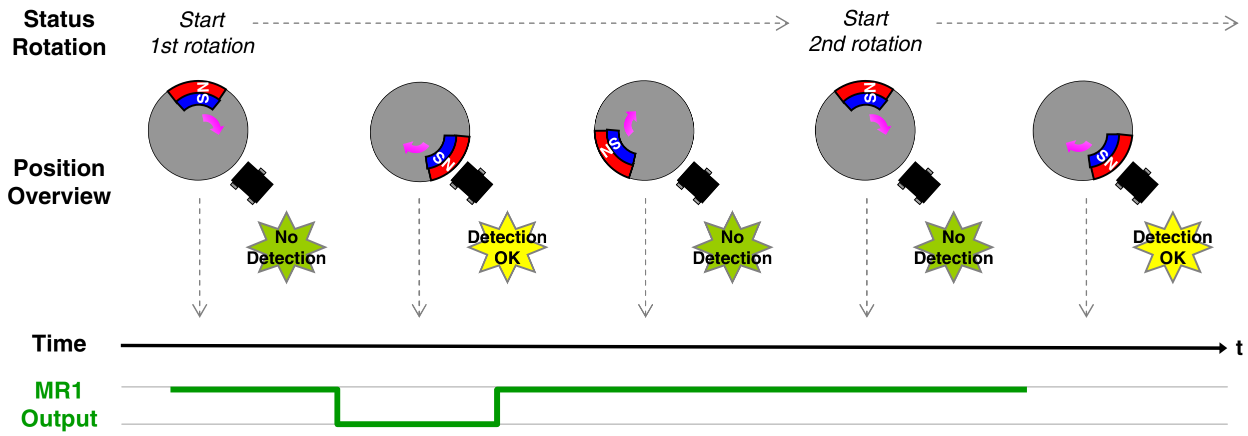

Figure 4-3 MR sensor used for rotation detection

Evaluation of the rotation speed:

To know the rotation speed, the easiest way is to measure the time between two edges of the rotation of one of the sensor, this means the pulse width. The exact speed is not relevant: we can simplify the application by only calculating the number of CPU clocks and compare it to some level values calculating, to have an idea of the level speed.

For example, with only two levels of speed "Low" and "High":

- If we have: Pulse width measured > Pulse width level
  - Conclusion: the speed is considered as Low
- If we have: Pulse width measured < Pulse width level
  - Conclusion: the speed is considered as High

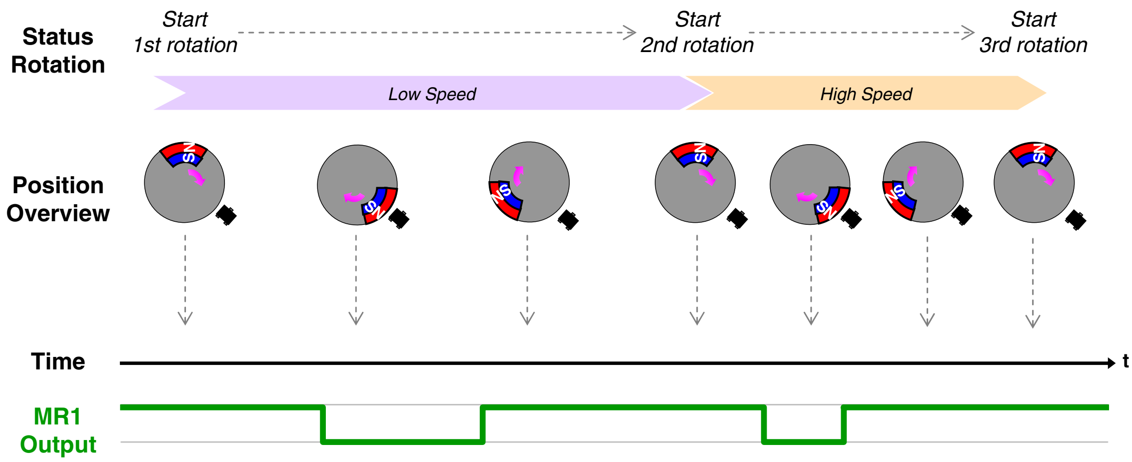

Figure 4-4 MR sensor output waveform depending of speed rotation

Detection of the direction:

In order to detect the direction of rotation, a second MR sensor has to be placed with a little angle near the first MR sensor.

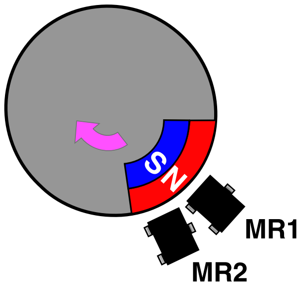

Figure 4-5 **Detection of direction with two MR sensors**

To know the direction, we have to observe:

- if the falling edge on MR sensor 1 is detect just before the falling edge on MR sensor 2, the direction of the rotation is “Clockwise”
- if the falling edge on MR sensor 2 is detected just before the falling edge on the MR sensor 1, the direction of the rotation is “Anticlockwise”

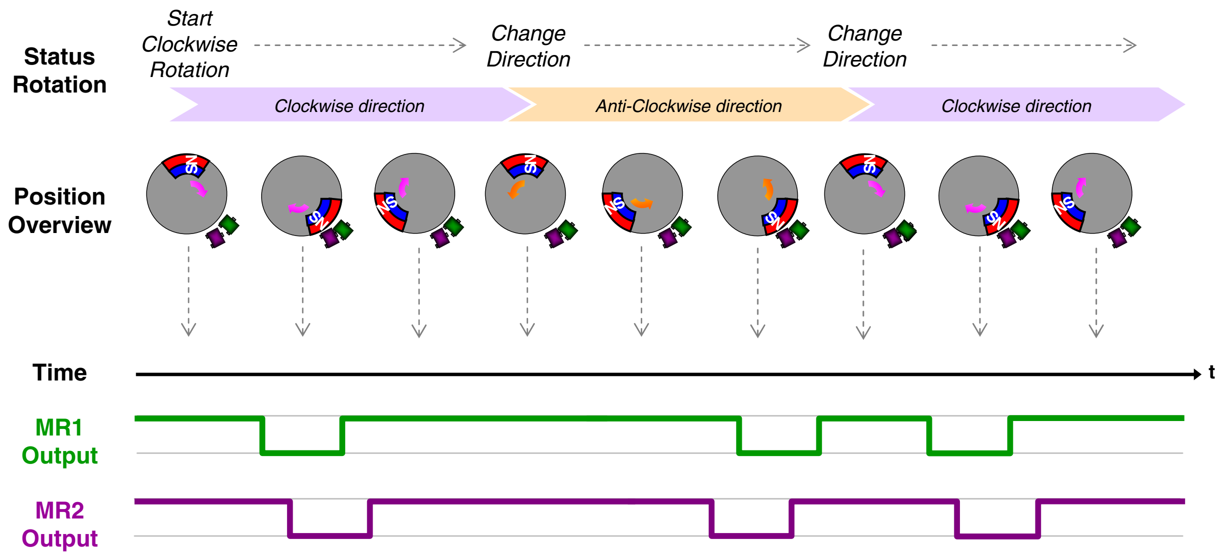

Figure 4-6 **MR sensor output waveforms depending of direction**

Depending of the application, it can be useful to put more than one magnet on the same wheel. So the pair of MR sensors will detect much more edges during each rotation. This only impacts the software that has to manage the same signal, but with higher speed and frequency.

4.2 Microcontroller Resources, Algorithm and Reference Design

In term of resources and connection to the microcontroller, this is still quite easy to handle:

The only requirement is to implement an algorithm that effectively can detect the rotation, the speed and the direction.

There are a lot of possibilities, and we will only give details of one method using two interruptible external pins to detect the rotation and its direction, and one timer for the speed measurement

To detect the rotation, any edge detection is enough.

To know the direction of rotation, it is required to use the outputs of the two MR sensors and determine which one of the two falling edges (or of the two rising edges) occurred first.

To determine the speed of the rotation, it is required to use a timer and determine the width of a low pulse created by one of the MR sensor. In most of the applications, a precise value of the rotation speed is not relevant: only an idea like “low speed” and “high speed” and additional step in between is enough. The value calculated by the timer is compared to some level values determined, to get an idea of the level speed.

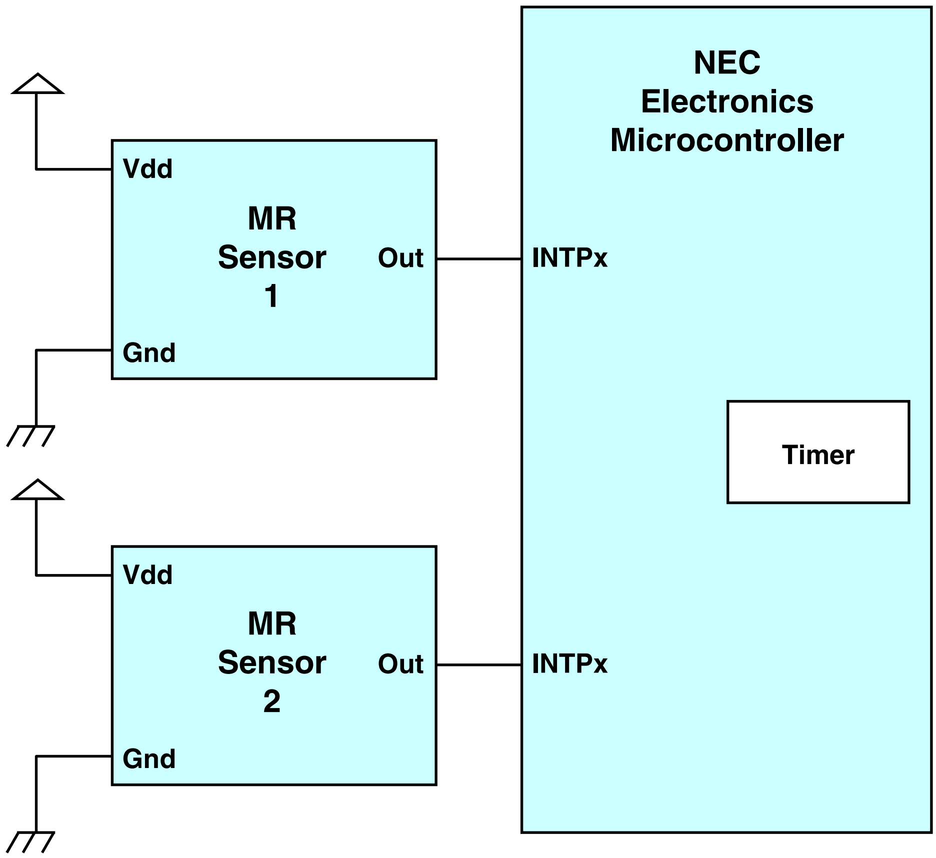

Figure 4-7 MR sensor typical connexion for jog dial application

4.3 Sample Program Example

A typical example of application is a counter which is incremented and decremented depending of the direction and speed of the rotation. To simplify the application, only two speed levels are implemented, just by the evaluation of timer overflow:

- If the overflow occurred, we qualified the speed as “low” and the counter is incremented/decremented 1 by 1.

- If the overflow of the timer does not occur the speed is qualified as “high” and the counter incremented/decremented 10 by 10.

Depending oin the rotation direction:

- LED1 is ON when anticlockwise rotation occurred, and counter is decremented
- LED2 is ON when Clockwise rotation occurred, and counter is incremented

This sample application was developed with NEC microcontroller 78K0R/KE3-L (µPD78F1009GB). The MR sensors chosen are 2 MRSS23E(E) powered with the same supply voltage of the microcontroller (3V). The complete software is given as reference in "*chapter 6.2*".

1. Contents of initial settings “Low_Level_init”
   - Selecting the 20MHz external resonator as the system clock source
   - Setting I/O ports P76 & P77: outputs for LED1 & LED2
   - Setting 2 external interrupt ports INTP1 & INTP2: detection of rising and falling edges on outputs of MR sensor 1 and 2.
   - Setting Timer Array Unit channel 0 as interval mode counter

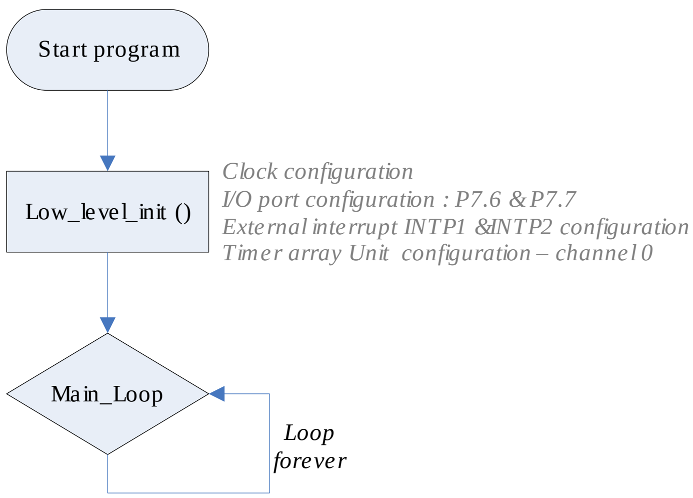

Figure 4-8 Main processing overview flowchart

2. Contents of main processing operation done by different interrupt
   - Detection of any edges to detect the rotation by interrupt
   - Evaluation of the edge occurred to detect the direction of the rotation
   - The speed value is determined to increment and decrement the counter with the timer:
     - If overflow of the timer occurred before the next edge is detected, we consider the speed as low, and increment/ decrement by 1.
     - If overflow of the timer did not occur before the next edge is detected, we consider the speed as high, and increment/decrement by 10.

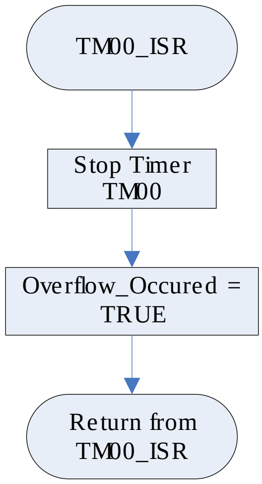

Figure 4-9 TM00 interrupt flowchart

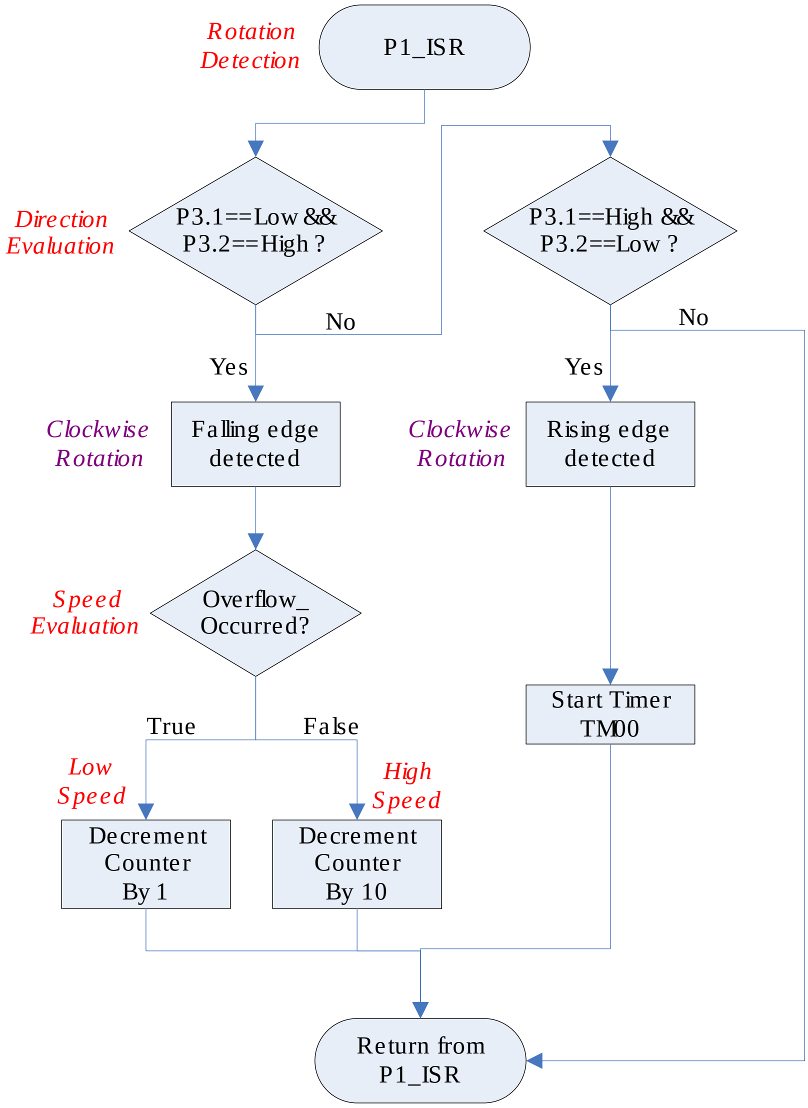

Figure 4-10 INTP1 interrupt flowchart

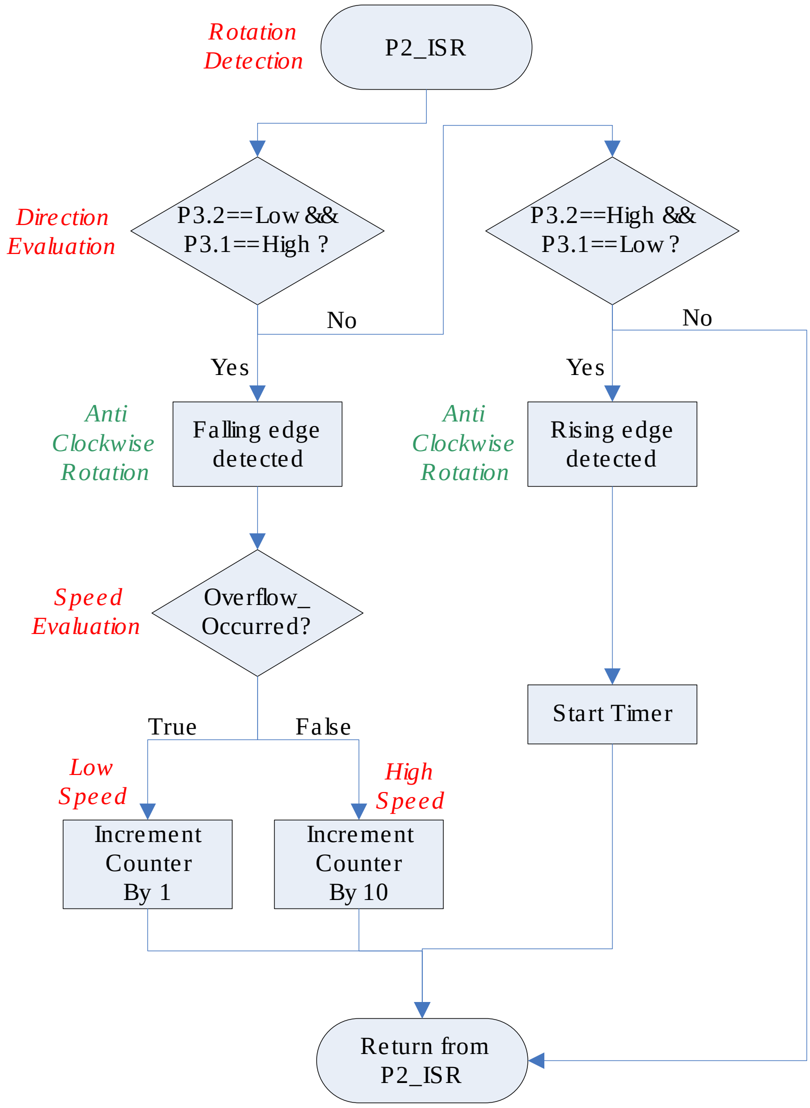

Figure 4-11 INTP2 interrupt flowchart

Chapter 5 Conclusion

This application note explains the basics of NEC Electronics MR Sensor, and shows two concrete application examples where MR sensor can be used: Open close detection and jog dial.

This shows how easy it is to use MR sensor. Associated with a microcontroller like NEC Electronics microcontroller such a device becomes a very smart application, and opens the door to multiple and innovative applications.

Due to its features, reliability and robustness, the MR sensor is the ideal candidate to replace Hall Sensor or Reed Switch Relay.

For more information, please visit our website *www.eu.necel.com*

Chapter 6 Appendix

6.1 Software Listings: Open/Close Detection

```
/*==========================================================================*/
/* Environment:                                                             */
/*        Device:          uPD78F1009                                       */
/*        Target Hardware: QB-78K0RKE3L-TB                                  */
/*        IDE:             IAR Systems - Embedded Workbench for 78K V4.xx   */
/*==========================================================================*/

/*==========================================================================*/
/* Module : main.c                                                          */
/* Purpose: main routine of sample                                          */
/*==========================================================================*/

/*==========================================================================*/
/* Includes                                                                 */
/*==========================================================================*/
/* IAR Systems Includes */
#include <intrinsics.h>
#include <io78f1009_64.h>
#include <io78f1009_64_ext.h>

/* Application Includes */
#include <types.h>
#include <macros.h>
#include <globals.h>


/*==========================================================================*/
/* Functions                                                                */
/*==========================================================================*/
void main (void)
{
     LED1 = OFF;
     __enable_interrupt();         /* enable interrupts */

     while(1)  {}
}



/*==========================================================================*/
/* Module: low_level_initialization.c                                       */
/* Purpose: basic microcontroller initialization                           */
/*==========================================================================*/

/*==========================================================================*/
/* Includes                                                                 */
/*==========================================================================*/
/* IAR Systems Includes */
#include <intrinsics.h>
#include <io78f1009_64.h>
#include <io78f1009_64_ext.h>

/* Application Includes */
#include <types.h>
#include <macros.h>
#include <low_level_initialization.h>


/*==========================================================================*/
/* Constants                                                                */
/*==========================================================================*/
```

```
/* Option Byte Definition */
/* watchdog disabled, LVI enabled, OCD interface enabled */
__root __far const UCHAR OptionByte[OPT_BYTES_SIZE] @ 0x00C0 = {
       WATCHDOG_DISABLED, LVI_ENABLED_20MHZ, RESERVED_FF, OCD_ENABLED,};

/* Security Byte Definition */
__root __far const UCHAR SecuIDCode[SECU_ID_SIZE]   @ 0x00C4 = {
       0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu };

/*==========================================================================*/
/* Functions                                                                */
/*==========================================================================*/
#pragma diag_suppress=Pm011
int __low_level_init (void)
#pragma diag_default=Pm011
{

    /* ---------------------------------------------------------------------*/
    /* Clock Configuration                                                  */
    /* ---------------------------------------------------------------------*/
    /* Set clock operation mode */
    CMC   = X1_X2_RES | XT1_XT2_INP | LOW_POWER_OSC |FX_10_20;
                             /* XT1 and XT2 pin in input port mode */
                             /* X1  and X2 pin in crystal resonator mode */
                             /* low power setting osc. margin 50KB */
                             /* 10MHz < fx <= 20MHz */

    OSTS  = STAB_TIME_MAX;   /* set oscillation stabilization time */
    OSMC  = FREQ_HIGH;       /* set speed mode: fx > 10MHz */

    /* Start X1 oscillator operation */
    MSTOP = 0u;              /* internal high-speed oscillator operating */

    /* Check oscillation stabilization time status */
    while(OSTC < OSC_STAB_MAX)  {
      __no_operation();      /* wait until X1 clock stabilization time has been elapsed  */
                             /* other initializations can be done */
    }
    /* Switch CPU clock to X1 oscillator */
    MCM0 = TRUE;
    while(MCS != TRUE)  {
        __no_operation();     /* wait until CPU and peripherals operate with fX1 clock */
    }

    /* Change clock generator setting, if necessary */
    CKC &= FREQ_OSC;          /* operating frequency f = fx */
                              /* From here onwards the X1 oscillator is supplied to the CPU */

    /* Stop the internal high-speed oscillator operation */
    HIOSTOP = TRUE;

    /* Stop the XT1 oscillator operation */
    XTSTOP  = TRUE;

    /* ---------------------------------------------------------------------*/
    /* I/O Port Configuration                                               */
    /* ---------------------------------------------------------------------*/
    P7_bit.no6  = FALSE;       /* clear port register P7.6 */
    PM7_bit.no6 = OUTPUT;      /* use port P7.6 as output */


    /* ---------------------------------------------------------------------*/
    /* External Interrupt Configuration                                     */
    /* ---------------------------------------------------------------------*/
    PU12_bit.no0  = TRUE;       /* enable pull-up resistor  */
    EGN0_bit.no0  = TRUE;       /* falling edge detection */
    EGP0_bit.no0  = TRUE;       /* rising edge detection  */
    PIF0          = FALSE;      /* clear interrupt request flag */
    PMK0          = IRQ_ENABLE; /* enable ext. INTP0 interrupt */

    return (int)(ENABLE_VAR_INIT);
}
```

```
/*==========================================================================*/
/* Module : interrupt.c                                                     */
/* Purpose: interrupt service routines                                      */
/*==========================================================================*/

/*==========================================================================*/
/* Includes                                                                 */
/*==========================================================================*/
/* IAR Systems Includes */
#include <intrinsics.h>
#include <io78f1009_64.h>

/* Application Includes */
#include <globals.h>


/*==========================================================================*/
/* Prototypes                                                               */
/*==========================================================================*/
static __interrupt void P0_isr   (void);


/*==========================================================================*/
/* Functions                                                                */
/*==========================================================================*/
/* Interrupt service routine of external interrupt 0 for MR sensor 1 */
#pragma vector=INTP0_vect
static __interrupt void P0_isr (void)
{
    if (P12_bit.no0 == HIGH ){          /* OPEN position detection */
      LED1 = ON;
    }
    else if  (P12_bit.no0 == LOW ) {    /* CLOSE position detection */
      LED1 = OFF;
    }
    else{ }
}
```

6.2 Software Listings: Rotation Application

```
/*==========================================================================*/
/* Environment:                                                             */
/*       Device:          uPD78F1009                                        */
/*       Target Hardware: QB-78K0RKE3L-TB                                   */
/*       IDE:             IAR Systems - Embedded Workbench for 78K V4.xx    */
/*==========================================================================*/

/*==========================================================================*/
/* Module : main.c                                                         */
/* Purpose: main routine of sample                                          */
/*==========================================================================*/

/*==========================================================================*/
/* Includes                                                                 */
/*==========================================================================*/
/* IAR Systems Includes */
#include <intrinsics.h>
#include <io78f1009_64.h>
#include <io78f1009_64_ext.h>

/* Application Includes */
#include <types.h>
#include <macros.h>
#include <globals.h>
```

```
/*==========================================================================*/
/* Functions                                                                */
/*==========================================================================*/
void main (void)
{
  overflow = FALSE;
  LED1 = OFF;
  LED2 = OFF;
   __enable_interrupt();          /* enable interrupts */

  while(1)  {}
}



/*==========================================================================*/
/* Module : low_level_initialization.c                                      */
/* Purpose: basic microcontroller initialization                            */
/*==========================================================================*/

/*==========================================================================*/
/* Includes                                                                 */
/*==========================================================================*/
/* IAR Systems Includes */
#include <intrinsics.h>
#include <io78f1009_64.h>
#include <io78f1009_64_ext.h>


/* Application Includes */
#include <types.h>
#include <macros.h>
#include <low_level_initialization.h>

/*==========================================================================*/
/* Constants                                                                */
/*==========================================================================*/
/* Option Byte Definition */
/* watchdog disabled, LVI enabled, OCD interface enabled */
__root __far const UCHAR OptionByte[OPT_BYTES_SIZE] @ 0x00C0 = {
       WATCHDOG_DISABLED, LVI_ENABLED_20MHZ, RESERVED_FF, OCD_ENABLED,};

/* Security Byte Definition */
__root __far const UCHAR SecuIDCode[SECU_ID_SIZE]   @ 0x00C4 = {
       0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu, 0xffu };



/*==========================================================================*/
/* Functions                                                                */
/*==========================================================================*/
#pragma diag_suppress=Pm011
int __low_level_init (void)
#pragma diag_default=Pm011
{

    /* -------------------------------------------------------------------*/
    /* Clock Configuration                                                */
    /* -------------------------------------------------------------------*/
    /* Set clock operation mode */
    CMC   = X1_X2_RES | XT1_XT2_INP | LOW_POWER_OSC |FX_10_20;
                         /* XT1 and XT2 pin in input port mode */
                         /* X1  and X2 pin in crystal resonator mode */
                         /* low power setting osc. margin 50KB */
                         /* 10MHz < fx <= 20MHz */

    OSTS  = STAB_TIME_MAX;   /* set oscillation stabilization time */
    OSMC  = FREQ_HIGH;       /* set speed mode: fx > 10MHz */

    /* Start X1 oscillator operation */
    MSTOP = 0u;              /* internal high-speed oscillator operating */

    /* Check oscillation stabilization time status */
```

```
    while(OSTC < OSC_STAB_MAX)  {
      __no_operation();      /* wait until X1 clock stabilization time has been elapsed */
                             /* other initializations can be done */
    }

    /* Switch CPU clock to X1 oscillator */
    MCM0 = TRUE;
    while(MCS != TRUE)  {
        __no_operation();     /* wait until CPU and peripherals operate with fX1 clock */
    }

    /* Change clock generator setting, if necessary */
    CKC &= FREQ_OSC;         /* operating frequency f = fx  */
                             /* From here onwards the X1 oscillator is supplied to the CPU */

    /* Stop the internal high-speed oscillator operation */
    HIOSTOP = TRUE;

    /* Stop the XT1 oscillator operation */
    XTSTOP  = TRUE;

    /* ---------------------------------------------------------------------*/
    /* I/O Port Configuration                                               */
    /* ---------------------------------------------------------------------*/
    P7_bit.no6  = FALSE;                /* clear port register P7.6  */
    P7_bit.no7  = FALSE;                /* clear port register P7.7  */
    PM7_bit.no6 = OUTPUT;               /* use port P7.6 as output   */
    PM7_bit.no7 = OUTPUT;               /* use port P7.6 as output   */

    /* ---------------------------------------------------------------------*/
    /* Timer Configuration                                                  */
    /* ---------------------------------------------------------------------*/
    PER2_bit.no0 = TRUE;         /* switch on timer arry unit input clock */
    TPS0 = CK00_FREQ_CPU_2048;   /* select TAU clock frequency CK00: fx/2048 */

    TMR00 = 0x0000u;        /* set timer mode TAU channel CK00 */
                            /* - operation clock set by PRS register */
                            /* - clock specified by CKS0 bit (bit 15) */
                            /* - slave channel  */
                            /* - software start trigger */
                            /* - interval timer mode   */

    TDR00 = 780u;           /* set 80ms interval time */
                            /* TDR00 = (10ms * 20MHz / 2048) -1 */

    TOE0 &= 0xFEu;
    TO0  &= 0xFEu;
    TOM0 &= 0xFEu;
    TOL0 &= 0xFEu;
    TMIF00 = FALSE;        /* clear TAU ch-0 interrupt request flag */
    TMMK00 = IRQ_ENABLE;   /* enable TAU ch-0 interrupt */
    TS0 |= 0x01u;          /* start timer */

    /* ---------------------------------------------------------------------*/
    /* External Interrupt Configuration                                     */
    /* ---------------------------------------------------------------------*/
    /* INTP2 settings for MR Sensor 1 */
    PU3_bit.no2  = TRUE;                /* enable pull-up resistor */
    EGN0_bit.no2  = TRUE;               /* falling edge detection */
    EGP0_bit.no2  = TRUE;               /* rising edge detection  */
    PIF2 = FALSE;                       /* clear interrupt request flag */
    PMK2 = IRQ_ENABLE;                  /* enable ext. INTP2 interrupt */

    /* INTP1 settings for MR Sensor 2 */
    PU3_bit.no1  = TRUE;                /* enable pull-up resistor */
    EGN0_bit.no1  = TRUE;               /* falling edge detection */
    EGP0_bit.no1  = TRUE;               /* rising edge detection  */
    PIF1          = FALSE;              /* clear interrupt request flag */
    PMK1          = IRQ_ENABLE;         /* enable ext. INTP0 interrupt */

    return (int)(ENABLE_VAR_INIT);
}
```

```
/*==========================================================================*/
/* Module : interrupt.c                                                     */
/* Purpose: interrupt service routines                                      */
/*==========================================================================*/

/*==========================================================================*/
/* Includes                                                                 */
/*==========================================================================*/
/* IAR Systems Includes */
#include <intrinsics.h>
#include <io78f1009_64.h>
#include <io78f1009_64_ext.h>

/* Application Includes */
#include <globals.h>


/*==========================================================================*/
/* Prototypes                                                               */
/*==========================================================================*/
static __interrupt void TM00_isr (void);
static __interrupt void P1_isr   (void);
static __interrupt void P2_isr   (void);


/*==========================================================================*/
/* Functions                                                                */
/*==========================================================================*/
/* Interrupt service routine of timer  */
#pragma vector=INTTM00_vect
static __interrupt void TM00_isr (void)
{
   TT0 |= 0x01u;        /* stop timer  */
   overflow = TRUE;
}

/* Interrupt service routine of external interrupt 1 for MR sensor 1 */
#pragma vector=INTP1_vect
static __interrupt void P1_isr (void)
{
    /* detection of Anti Clockwise rotation */
    if ( (P3_bit.no1 == LOW) && (P3_bit.no2 == HIGH) ) {
          LED2 = OFF;
          if (overflow == TRUE) {
              if  (numero!=0x00u) { numero--; }
          }
          else {
              TT0 |= 0x01u;        /* stop timer  */
              if  (numero<0x0Au) {
                  numero = 0x00u;
              }
              else {
                  numero = numero - 0xAu;
              }
        }
    }

    else  if ( (P3_bit.no1 == HIGH) && (P3_bit.no2 == LOW) ) {
            LED2 = ON;
            TS0  |= 0x01u;        /* start timer  */
            overflow = FALSE;
    }

    else {}

}


/* Interrupt service routine of external interrupt 2 for MR sensor 2 */
#pragma vector=INTP2_vect
static __interrupt void P2_isr (void)
{
```

```
   /* detection of Clockwise rotation  */
   if ( (P3_bit.no2 == LOW) && (P3_bit.no1 == HIGH) ){
            LED1 = OFF;
            if (overflow == TRUE) {
                if  (numero!=0xFFu) { numero++; }
            }
            else {
                TT0 |= 0x01u;   /* stop timer  */
                if  (numero>0xF5u) {
                      numero = 0xFFu;
                }
                else {
                      numero = numero + 0xAu;
                }
            }
    }

    else if ( (P3_bit.no2 == HIGH) && (P3_bit.no1 == LOW) ){
            LED1 = ON;
            TS0  |= 0x01u;    /* start timer  */
            overflow = FALSE;
    }
    else {}

}
```