**Topic:** To study position encoder and its working

**Pre-requisite knowledge:** Interrupt concept and basic motion control

**Components required:** Firebird V with 8051 adapter board

**Basics:**

* **What are position encoders?**

Position Encoder sensors are used to find position of the wheel. It consists of IR LED and Photodiode mounted facing each other enclosed in plastic body. It gives position and velocity feedback to the robot.

* **General Working:** When light emitted by the IR LED is blocked because of alternating slots of the encoder disc logic level of the photo diode changes. This change in the logic level can be sensed by the microcontroller or by discrete hardware. This sensor is used to give position feedback to the robot.
* **In Firebird V, position encoder sensor used is MOC7811.**



* **Working in Firebird V:**

It is used in closed loop. Position encoder consists of optical encoder and slotted disc assembly. When this slotted disc rotates it cuts IR illumination from IR LED alternately due to which photo transistor gives square pulse train as output. The pulse count of square wave output indicates position and time period of square wave output indicates velocity.



**Fig: DC geared motors and position encoders**



**Fig: Position encoder assembly**

**Note:** On the P89V51RD2 microcontroller board for Fire Bird V robot left encoder is connected to Interrupt 1 while right encoder is connected to Timer 1 input pin of the P89V51RD2 microcontroller. To connect interrupt1 with the position encoder, connect the jumper as shown in fig.



Now the students must be clear about the position encoders, how they work and their position on the adapter board.

The next step is to program them to control the position of the robot. For this they should first know how to calculate the **resolution** of position encoder.

**Resolution:** It is defined as the smallest change in input (here distance) which can be detected by the position encoder sensor.

* **When motion of the robot is linear, the resolution is in mm whereas when motion of the robot is angular, the resolution is in degrees.**

**For example –**

Let’s say the robot is moving forward and wheel diameter is 5.1cm and Number slots on the encoder disc are 30, then

Wheel circumference = 5.1cm \* 3.14

= 16.014cm

= 160.14mm

Position encoder resolution = Wheel circumference/ number of slots

= 160.14 mm/ 30

= 5.338 mm/ pulse.

Now when robot is moving with one wheel rotating in clockwise direction and another wheel in anticlockwise direction i.e. the robot is taking turn then the resolution can be calculated as,

Distance between Wheels = 15cm

Radius of Circle formed in 360 degree rotation of Robot = Distance between Wheels / 2

= 7.5 cm

Distance Covered by Robot in 360 degree Rotation = Circumference of Circle traced

= 2 x 7.5 x 3.14

= 47.1 cm or 471mm

Number of wheel rotations of in 360 degree rotation of robot = Circumference of Traced Circle / Circumference of Wheel

= 471 / 160.14

= 2.941

Total pulses in 360 degree rotation of robot = Number of slots on the encoder disc x Number of wheel rotations of in 3600 rotation of robot

= 30 x 2.941

= 88.23 (approximately 88)

Position Encoder Resolution in Degrees = 360 / 88

= 4.091 degrees per count

Now students know that the resolution in case of linear motion is 5.44 mm/pulse and the concept of position control using interrupts (covered in previous section) so now we’ll program it to control position of the robot.

In the previous section (interrupts), we learned about an unsigned integer variable in ISR which increments every time an interrupt occurs. This count is used to calculate the actual distance travelled by the robot using the following formula.

Distance (in mm) = count \* 5.338

Or

Required count = distance/5.338

By continuously comparing the current ‘count’ to the ‘required count’ as obtained above, position of the robot can be controlled.

**Using Interrupts:**

Interrupts need to be initialized before they can be used. Initializing interrupts involves programming the following registers –

**IEN0 = 0x84; Interrupt enable register 0**

**TCON = 0x04; Timer/counter control register:**

Now next step after interrupt initialization is to define ISR (Interrupt Service Routine) whose syntax differs with different compilers. Syntax for Keil is:

**void int1\_isr(void)interrupt 2**

Main logic to be used in algorithm is that it counts pulses from left shaft encoder, checks if a particular distance has been covered by the robot, and stops the robot after this distance is travelled.

Now the **algorithm** to move robot for a particular distance can be written as follows:

1. Define variable for particular distance, required shaft count and left shaft count.
2. Required shaft count = distance/ 5.338
3. Initialize interrupts
4. Define ISR
5. Repeat while (left shaft count<required shaft count)
   1. Move forward
6. End repeat
7. End

* **Code for the above is given in experiments folder.**

Similarly algorithm for angular movement of robot can also be written.

**Exercise:**

1. Program the robot to move 50 cm forward
2. Program the robot to take 90 degrees turn
3. Program the robot to trace square of side 50 cms.