In the following two problems a system to measure the angle, distance, and speed of a rotating disk will be designed. The disk can rotate clockwise and counterclockwise, and can change direction at any time. The system need not determine angle and distance until the disk has made one full rotation, but then must update them as frequently as possible.

The disk has space for three rings of marks; the transducers, conditioning circuits, and interfaces for the mark readers are provided (they are not part of the solution). The first problem is to determine the placement of the marks, the second problem is to write the interface routine (actually, an interrupt handler).

**Problem 1:** Show how to place marks on the three rings to achieve maximum precision, remembering that rotation direction can change; each ring can have up to 1024 marks. Clearly show how the marks are positioned. For convenience, the mark rings can be shown as linear tracks. *Hint: Straightforward adoption of a coded displacement transducer presented in class would yield a precision of* $\frac{360^\circ}{1024}$*, a better answer would have a precision of* $\frac{360^\circ}{4\times 1024}$*, a much better answer would have a precision of* $\frac{360^\circ}{6\times 1024}$*. For the much better answer, think gray codes.*

**Problem 2:** Write a handler routine to determine angle, distance, and speed, as described below. The presence of a mark under the transducers is determined by calling the `readInterface(ring)` function, where integer `ring` is the ring number to be read, either 0, 1, or 2. The function returns an integer; 0 indicates that there is not a mark under the transducer, a 1 indicates that there is.

Each time the state of a transducer changes (a mark moves under the transducer or a mark moves out from under the transducer) the handler routine, `disk_routine`, will be called. When called, that routine should update (if appropriate) the variables `disk_angle`, `disk_distance`, and `disk_speed`. Variable `disk_angle` should contain the disk angle, in radians, with respect to a reference point (determined by the marks). Variable `disk_distance` should contain the total angular distance rotated, in radians. For example, if the disk rotates clockwise $\pi$ radians and counterclockwise $2\pi$ radians, then the distance would be $3\pi$. (Because the disk can change direction at any time, the accuracy of the distance is limited by the precision to which angle can be determined.) Finally, `disk_speed` should contain the angular velocity in radians per second, based on the two most recent transducer state changes.

A function `get_current_time()` is available, it returns a double-precision number, the number of seconds in the Unix epoch (since 1 January 1970, 00:00 UTC), down to microsecond precision (it is not an integer).

The solution can be based on the code shell below:

```
#define N (1<<10)   /* Number of marks per ring. */
/* Below, radians per mark. Note: M_PI defined in math.h. */
#define DELTA_THETA ( 2.0 * M_PI / N )

static int last_mark0, last_mark1, last_mark2;
/* direction: -1 Counterclockwise, +1 clockwise, 0 not initialized. */
static int direction = 0;
static double last_time;
double disk_angle, disk_distance, disk_speed;

/* Called each time transducer output changes. */
void disk_routine()
{
  int this_mark0 = readInterface(0);
  int this_mark1 = readInterface(1);
  int this_mark2 = readInterface(2);
  double now = get_current_time();

  if( direction == 0 )
    {
      /* If at zero-radian angle, initialize. */
      /* Solution code here. */




    }
  else
    {
      /* Compute direction, angle, etc. */
      /* Solution code here. */


      disk_angle += /* Answer here. */ ;
      disk_distance += /* Answer here. */ ;
      disk_speed = /* Answer here. */ ;
    }

  last_time = now;
  last_mark0 = this_mark0; last_mark1 = this_mark1; last_mark2 = this_mark2;

}
```