```
@0
new

@0
singleline(1)
separated(1)
channel = 1
direction = -1
max_steps = 3000
detent_width = 110
filter_space = 310
detent_mask = 0
datum_mask = 1
dim filter_pos[2]

@0
Proc zero1() {
 set_up(1)
 if(find_home())
  print "0"
 else
  print "1"
}

@0
Proc zero2() {
 set_up(2)
 if(find_home())
  print "0"
 else
  print "1"
}

@0
proc filter() {
 set_up(%1)
 b = %2
 // determine direction in which to go
 if (b != Filter_pos[channel]) {
  if (b > Filter_pos[channel]) {
   c = b - Filter_pos[channel] - 1
   f =  max_steps * direction
  } else {
   c = Filter_pos[channel] - b - 1
   f =  0
  }

  // start motion
  move(channel, f)

  // wait for this detent to release
  while (moving(channel) && (de_bounce(detent_mask) == 0) {}

  // count c detents
  for (i = 0; i < c; i++) {
   while (moving(channel) && (de_bounce(detent_mask) == 1) {}
   while (moving(channel) && (de_bounce(detent_mask) == 0) {}
  }

  // capture required filter
  if (find_detent()) {
   filter_pos[channel] = b
   print "0"
  } else
   print "1"
 } else
  print "0"
}

@0
func find_home() {
 if (find_datum()) {
  move(channel, filter_space * direction)
  if (find_detent()) {
   datum(channel)
   filter_pos[channel] = 1
   return 1
  }
 }
 return 0
}

@0
func find_datum() {
 // move away if we are at datum already
 if (de_bounce(datum_mask) == 0) {
  move(channel, where(channel) + (500 * direction))
  while(moving(channel)){}
 }

 // move backwards full throw to datum park
 move(channel, max_steps * direction *  -1)

 // sense datum and rough stop
 while(moving(channel)) {
  if(de_bounce(datum_mask) == 0) {
   halt(channel)
   while(moving(channel)) {}
   return 1
  }
 }
 return 0
}

@0
func find_detent() {
 // make sure were moving
 if (moving(channel) == 0)
  return 0

 // determine direction of motion
 x1 = where(channel)
 d = 0
 while ((where(channel) == x) && moving(channel)) {}
 if (((where(channel) - x) * direction) < 0)
  d = 1

 // make sure we are not in a datum hole
 while(moving(channel) && (de_bounce(detent_mask) == 0)) {}

 // setup for search
 s = 1
 x1 = 0
 x2 = 0

 // measure detent width
 while(moving(channel)) {
  if (de_bounce(detent_mask) != s) {
   x = where(channel)
   if (s) {
    x1 = x
    s = 0
   } else {
    x2 = x
    halt(channel)
    s = 1
   }
  }
 }

 // if we have approached backwards, go to first edge
 if (d) {
  move(channel, x1)
  while(moving(channel)) {}
 }

 // step into calculated middle of detent
 x = (x1 + x2) / 2
 move(channel, x)
 while(moving(channel)) {}

 // check detent is home
 if (de_bounce(detent_mask))
  return 0

 return 1
}

@0
proc setup() {
 if (%1 < 2) {
 // filter wheel 1
  channel = 1
  direction = 1
  detent_mask = 0
  datum_mask = 1
 } else {
  // filter wheel 2
  channel = 0
  direction = -1
  detent_mask = 2
  datum_mask = 3
 }
}

@0
func de_bounce() {
 // simple majority vote on three
 q = 0
 for (i = 0; i < 3; i++) {
  q += in(%1)
  wait(5)
 }
 if (q > 1)
  return 1
 return 0
}

@1
new

@1
singleline(1)
separated(1)

proc zerofoc() {
 //move away from datum if already in region
 if (in(4)==0){
  move(0,1800)
  while(moving()){}
 }

 move(0,-5500)
 while(moving()){
  if (in(4)==0){
   halt(0)
   x=where(0)
   for(i=1;in(4)==0;i++){
    move(0,x+i)
    while(moving()){}
   }
   print"0"
   datum(1)
  }
 }
}

@1
proc focus(){
move(0,%1)
}
```