Program1_ball_drop.lsc

```
/*
/0LOGBOOK
0

*/
program test {

      #include <RCX2.h>
      #include <RCX2MLT.h>
      #include <RCX2Sounds.h>
      #include <RCX2Def.h>
      var C_counts = 0
      sensor light1 on 1
      light1 is light as percent
      event lBrite_light1EventHigh when light1.high


      sensor light2 on 2
      light2 is light as percent
      event lBrite_light2EventHigh when light2 > 30

      sensor rotation3 on 3
      rotation3 is rotation as angle
      event equal_rotation3EventEqual when rotation3 = 1

      event greater_C_countsEventHigh when C_counts > 2100

      event greater_C_countsEventHigh0 when C_counts > 2000

      macro DROP_BALL {
            counter1 = (light2*10)
            display counter1:1
            direction [  ] [ A ]
            on [ A  ]  for 100
            direction [ A  ] [ ]
            on [ A  ]  for 100
      }

      main {
            ext InterfaceType "kFreestyle"
            rcx_ClearTimers
            bbs_GlobalReset([A B C])
            start LightWatcher0
            start LightWatcher1
            start RotationWatcher2
            rcx_Priority( 8)
            trigger lBrite_light1EventHigh
            trigger lBrite_light2EventHigh
            trigger equal_rotation3EventEqual
            try {
                  direction [ C  ] [ ]
                  C_counts = 0
                  rcx_Calibrate(4,4)
                  clear Rotation3
                  on [ C  ]
            } retry on fail
            try {
                  repeat {
                        C_counts = (rotation3*10)
                        C_counts = abs(C_counts)
                  } until greater_C_countsEventHigh
            } retry on fail
            try {
                  float [ C  ]
            } retry on fail
            try {
                  DROP_BALL
            } retry on fail
            try {
                  direction [  ] [ C ]
                  C_counts = 0
                  rcx_Calibrate(4,4)
                  clear Rotation3
                  on [ C  ]
            } retry on fail
            try {
                  repeat {
                        C_counts = (rotation3*10)
                        C_counts = abs(C_counts)
                  } until greater_C_countsEventHigh0
            } retry on fail
            try {
                  float [ C  ]
                  stop tasks
            } retry on fail
      }


      watcher LightWatcher0 monitor lBrite_light1EventHigh
      {
            rcx_Priority( 4 )
            try {
            } restart on fail
      } restart on event
      watcher LightWatcher1 monitor lBrite_light2EventHigh
      {
            rcx_Priority( 4 )
            try {
            } restart on fail
      } restart on event
      watcher RotationWatcher2 monitor equal_rotation3EventEqual
      {
            rcx_Priority( 5 )
            try {
            } restart on fail
      } restart on event
}
```

Program2_server_food.lsc

```
/*
/0LOGBOOK
0

*/
program test {

      #include <RCX2.h>
```

```
#include <RCX2MLT.h>
#include <RCX2Sounds.h>
#include <RCX2Def.h>
var C_counter = 0
sensor light1 on 1
light1 is light as percent
event lBrite_light1EventHigh when light1 > 99

sensor light2 on 2
light2 is light as percent
event lDark_light2EventLow when light2.low


sensor rotation3 on 3
rotation3 is rotation as angle
event equal_rotation3EventEqual when rotation3 = 1

event greater_C_counterEventHigh when C_counter > 900

event greater_C_counterEventHigh0 when C_counter > 800


main {
      ext InterfaceType "kFreestyle"
      rcx_ClearTimers
      bbs_GlobalReset([A B C])
      start LightWatcher0
      start LightWatcher1
      start RotationWatcher2
      rcx_Priority( 8)
      trigger lBrite_light1EventHigh
      trigger lDark_light2EventLow
      trigger equal_rotation3EventEqual
      try {
            direction [ C  ] [ ]
            C_counter = 0
            rcx_Calibrate(4,4)
            clear Rotation3
            on [ C  ]
      } retry on fail
      try {
            repeat {
                  C_counter = (rotation3*10)
                  C_counter = abs(C_counter)
            } until greater_C_counterEventHigh
      } retry on fail
      try {
            off [ C  ]
            direction [  ] [ A ]
            on [ A  ]  for 100
            sound 4
            direction [  ] [ C ]
            power [ C  ] 8
            rcx_Calibrate(4,4)
            clear Rotation3
            C_counter = 0
            on [ C  ]
      } retry on fail
      try {
            repeat {
```

```
                        C_counter = (rotation3*10)
                        C_counter = abs(C_counter)
                  } until greater_C_counterEventHigh0
            } retry on fail
            try {
                  off [ C  ]
                  stop tasks
            } retry on fail
      }


      watcher LightWatcher0 monitor lBrite_light1EventHigh
      {
            rcx_Priority( 4 )
            try {
            } restart on fail
      } restart on event
      watcher LightWatcher1 monitor lDark_light2EventLow
      {
            rcx_Priority( 4 )
            try {
            } restart on fail
      } restart on event
      watcher RotationWatcher2 monitor equal_rotation3EventEqual
      {
            rcx_Priority( 5 )
            try {
            } restart on fail
      } restart on event
}
```

Program3_flag_gate_chairs2.lsc

```
program test {

      #include <RCX2.h>
      #include <RCX2MLT.h>
      #include <RCX2Sounds.h>
      #include <RCX2Def.h>
      var C_Counts = 0
      var B_Counts = 0
      var LS1_Bus_Color = 0
      var flag_Bus_Stop_Is = 0
      var turn_counts = 0
      var fwd_counts = 0
      var bwd_counts = 0
      var flag_search_chair2 = 0
      var turn2_counts = 0
      var fwd2_counts = 0
      var LS1_whiteFlag = 0
      var LS1_Difference = 0
      var LS1_background = 0
      var flag_BackgroundMode = 0
      var LS1_WeightedAverage = 0
      var tmp1 = 0
      var flag_search = 0
      var LS2_Field_Color = 0
      var skip_spaces = 0
      var Found_Red = 0
      sensor light2 on 2
```

```
light2 is light as percent
event lBrite_light2EventHigh when light2 > 99

event timer1_timer1Event when timer1 > 10
event equal_flag_searchEventEqual when flag_search = 30

sensor rotation3 on 3
rotation3 is rotation as angle
sensor light1 on 1
light1 is light as percent
event greater_C_CountsEventHigh when C_Counts > 700

macro FWD_CHAIR1 {
      C_Counts = 0
      direction [ C  ] [ ]
      on [ C  ]
      direction [ B  ] [ ]
      on [ B  ]  for 10
      while C_Counts < 1300 {
            C_Counts = (rotation3*10)
            C_Counts = abs(C_Counts)
      }
      off [ C  ]
}
macro PUSH_CHAIR1 {
      B_Counts = 0
      direction [  ] [ B ]
      on [ B  ]
      while B_Counts < 400 {
      }
      off [ B  ]
      B_Counts = 0
      direction [ B  ] [ ]
      on [ B  ]
      while B_Counts < 400 {
      }
      off [ B  ]
}
macro FIND_THE_BUS {
      if flag_search = 10{
            rcx_Calibrate(4,4)
            clear Rotation3
            C_Counts = 0
            direction [ C  ] [ ]
            on [ C  ]
            repeat {
                  C_Counts = (rotation3*10)
                  C_Counts = abs(C_Counts)
                  if flag_Bus_Stop_Is > 0{
                        C_Counts = 9990
                  }
                  else
                  {
                  }
                  if light1 > (LS1_Bus_Color / 10){
                        off [ C  ]
                        sound 3
                        flag_search = 20
                        flag_Bus_Stop_Is = 10
                        C_Counts = 9990
            }
            else
            {
            }
      } until greater_C_CountsEventHigh
      off [ C  ]
}
else
{
}
if flag_search = 10{
      rcx_Calibrate(4,4)
      clear Rotation3
      C_Counts = 0
      direction [ C  ] [ ]
      on [ C  ]
      repeat {
            if flag_Bus_Stop_Is > 0{
                  C_Counts = 9990
            }
            else
            {
            }
            C_Counts = (rotation3*10)
            C_Counts = abs(C_Counts)
            if light1 > (LS1_Bus_Color / 10){
                  off [ C  ]
                  sound 2
                  flag_search = 20
                  flag_Bus_Stop_Is = 20
                  C_Counts = 9990
            }
            else
            {
            }
      } until greater_C_CountsEventHigh
      off [ C  ]
}
else
{
}
if flag_search = 10{
      rcx_Calibrate(4,4)
      clear Rotation3
      C_Counts = 0
      direction [ C  ] [ ]
      on [ C  ]
      repeat {
            C_Counts = (rotation3*10)
            C_Counts = abs(C_Counts)
            if light1 > (LS1_Bus_Color / 10){
                  off [ C  ]
                  sound 4
                  flag_search = 20
                  flag_Bus_Stop_Is = 30
            }
            else
            {
            }
            if flag_Bus_Stop_Is > 0{
```

```
                        C_Counts = 9990
                  }
                  else
                  {
                  }
            } until greater_C_CountsEventHigh
            off [ C  ]
      }
      else
      {
      }
}
macro WACK_BUS_STOP {
      direction [  ] [ A ]
      on [ A  ]  for 40
      direction [ A  ] [ ]
      on [ A  ]  for 40
}
macro SEARCH_FOR_GATE {
      if flag_Bus_Stop_Is = 10{
            turn_counts = 220
            fwd_counts = 1230
      }
      else
      {
            if flag_Bus_Stop_Is = 20{
                  turn_counts = 370
                  fwd_counts = 460
            }
            else
            {
                  turn_counts = 560
                  fwd_counts = 200
            }
      }
      B_Counts = 0
      direction [  ] [ B ]
      on [ B  ]
      while B_Counts < turn_counts {
      }
      off [ B  ]
      B_Counts = 0
      C_Counts = 0
      rcx_Calibrate(4,4)
      clear Rotation3
      direction [ C  ] [ ]
      on [ C  ]
      while C_Counts < fwd_counts {
            C_Counts = (rotation3*10)
            C_Counts = abs(C_Counts)
      }
      off [ C  ]
}
macro WACK_GATE {
      off [ A B C  ]
      direction [  ] [ A ]
      on [ A  ]  for 100
      direction [ A  ] [ ]
      on [ A  ]  for 50
}
```

```
      macro SET_VALUES {
            comment (0, 0 ) "Iniitalize a set of distance and turn
parameters based on which flag was found."
            if flag_Bus_Stop_Is = 10{
                  bwd_counts = 0
                  turn_counts = 800
                  fwd_counts = 550
                  turn2_counts = 100
                  fwd2_counts = 750
                  skip_spaces = 200
            }
            else
            {
                  if flag_Bus_Stop_Is = 20{
                        bwd_counts = 0
                        turn_counts = 600
                        fwd_counts = 480
                        turn2_counts = 130
                        fwd2_counts = 750
                        skip_spaces = 250
                  }
                  else
                  {
                        bwd_counts = 0
                        turn_counts = 360
                        fwd_counts = 1420
                        turn2_counts = 160
                        fwd2_counts = 710
                        skip_spaces = 550
                  }
            }
      }
      macro FIND_RED_LINE {
            C_Counts = 0
            rcx_Calibrate(4,4)
            clear Rotation3
            direction [  ] [ C ]
            B_Counts = 0
            direction [  ] [ B ]
            comment (0, 0 ) "Rotate the robot by turn_counts"
            on [ B  ]
            while B_Counts < turn_counts {
            }
            off [ B  ]
            B_Counts = 0
            C_Counts = 0
            comment (0, 0 ) "Warning tone if the touch sensor happens to
be pressed."
            if light2 > 99{
                  sound 3
            }
            else
            {
            }
            rcx_Calibrate(4,4)
            clear Rotation3
            C_Counts = 0
            direction [ C  ] [ ]
            comment (0, 0 ) "Now move forward until we find the white
area of the basketball field#c just past the red line."
```

```
            on [ C  ]
            comment (0, 0 ) "Wiat to make sure we don#st hit the walkway
to the steps."
            while C_Counts < skip_spaces {
                  C_Counts = (rotation3*10)
                  C_Counts = abs(C_Counts)
            }
            Found_Red = 0
            while Found_Red < 10 {
                  if light2 < 98{
                        if light2 > (LS2_Field_Color / 10){
                              off [ C  ]
                              Found_Red = 11
                        }
                        else
                        {
                        }
                  }
                  else
                  {
                        on [ B  ]  for 5
                  }
            }
      }
      macro SEARCH_CHAIR2 {
            comment (0, 0 ) "Set the turn and forward counts based on
which flag was found."
            B_Counts = 0
            direction [  ] [ B ]
            comment (0, 0 ) "Rotate by turn2_counts to line up on the
chair."
            on [ B  ]
            while B_Counts < turn2_counts {
            }
            off [ B  ]
            B_Counts = 0
            C_Counts = 0
            rcx_Calibrate(4,4)
            clear Rotation3
            direction [ C  ] [ ]
            comment (0, 0 ) "Move up to the chair."
            on [ C  ]
            while C_Counts < fwd2_counts {
                  C_Counts = (rotation3*10)
                  C_Counts = abs(C_Counts)
            }
            off [ C  ]
      }
      macro PUSH_CHAIR2 {
            comment (0, 0 ) "Now push in that chair."
            B_Counts = 0
            direction [  ] [ B ]
            on [ B  ]
            while B_Counts < 400 {
            }
            off [ B  ]
            B_Counts = 0
            direction [ B  ] [ ]
            on [ B  ]
            while B_Counts < 400 {
            }
            off [ B  ]
      }
      macro TO_BASE {
            if flag_Bus_Stop_Is = 10{
                  fwd_counts = 850
                  turn2_counts = 300
                  fwd2_counts = 350
            }
            else
            {
                  if flag_Bus_Stop_Is = 20{
                        fwd_counts = 850
                        turn2_counts = 120
                        fwd2_counts = 500
                  }
                  else
                  {
                        fwd_counts = 850
                        turn2_counts = 120
                        fwd2_counts = 500
                  }
            }
            C_Counts = 0
            B_Counts = 0
            rcx_Calibrate(4,4)
            clear Rotation3
            direction [ C  ] [ ]
            on [ C  ]
            while C_Counts < fwd_counts {
                  C_Counts = (rotation3*10)
                  C_Counts = abs(C_Counts)
                  on [ B  ]
                  direction [  ] [ B ]
                  while B_Counts < 120 {
                  }
                  off [ B  ]
            }
      }
      event lBrite_light2EventHigh0 when light2 > (LS2_Field_Color / 10)


      main {
            ext InterfaceType "kFreestyle"
            rcx_ClearTimers
            bbs_GlobalReset([A B C])
            start LightWatcher0
            start TimerWatcher1
            start variableWatcher2
            rcx_Priority( 8)
            trigger lBrite_light2EventHigh
            trigger timer1_timer1Event
            trigger equal_flag_searchEventEqual
            try {
                  display flag_Bus_Stop_Is:1
                  flag_Bus_Stop_Is = 0
                  flag_search = 0
                  LS1_Bus_Color = (light1*10)
                  LS1_Bus_Color -= 40
                  LS2_Field_Color = (light2*10)
                  LS2_Field_Color -= 40
            } retry on fail
            try {
                  FWD_CHAIR1
            } retry on fail
            try {
                  PUSH_CHAIR1
            } retry on fail
            try {
                  flag_search = 10
            } retry on fail
            try {
                  FIND_THE_BUS
            } retry on fail
            try {
                  WACK_BUS_STOP
            } retry on fail
            try {
                  SEARCH_FOR_GATE
            } retry on fail
            try {
                  WACK_GATE
            } retry on fail
            try {
                  flag_search = 30
            } retry on fail
      }


      watcher LightWatcher0 monitor lBrite_light2EventHigh
      {
            rcx_Priority( 4 )
            try {
                  B_Counts += 10
            } restart on fail
      } restart on event
      watcher TimerWatcher1 monitor timer1_timer1Event
      {
            rcx_Priority( 8)
            try {
                  if flag_search = 10{
                        direction [ B  ] [ ]
                        on [ B  ]  for 10
                        clear Timer1
                  }
                  else
                  {
                  }
            } restart on fail
      } restart on event
      watcher variableWatcher2 monitor equal_flag_searchEventEqual
      {
            rcx_Priority( 2 )
            try {
                  SET_VALUES
                  FIND_RED_LINE
                  SEARCH_CHAIR2
                  PUSH_CHAIR2
                  TO_BASE
                  stop tasks
            } restart on fail
      } restart on event
      fragment ( 860,3390 ) {
            display C_Counts:1
      }
      fragment ( 940,1965 ) {
            off [ C  ]
            comment (0, 0 ) "Hunt for that white area."
            wait until lBrite_light2EventHigh0
      }
}
```

Program4_cd_glasses.lsc

```
program test {

      #include <RCX2.h>
      #include <RCX2MLT.h>
      #include <RCX2Sounds.h>
      #include <RCX2Def.h>

      main {
            ext InterfaceType "kFreestyle"
            rcx_ClearTimers
            bbs_GlobalReset([A B C])
            try {
                  power [ C  ] 8
                  direction [ C  ] [ ]
                  on [ C  ]  for 250
                  direction [  ] [ C ]
                  on [ C  ]  for 250
                  stop tasks
            } retry on fail
      }


}
```