Модуль rubiroMotors, моторы Lego EV3

В текущей версии библиотеки поддерживается два Lego-мотора - большой и средний. Отличия между моторами - в мощности и скорости вращения. Средний мотор более скоростной, но менее мощный, чем большой. Для среднего мотора используется класс **TEv3MediumMotor**, для большого - класс **TEv3LargeMotor**. Оба класса порождены от базового **TEv3TachoMotor**:

```
TEv3TachoMotor=class(TEv3Motor)
    function wait():TEv3TachoMotor; // ждет остановки мотора

    {
     Устанавливает способ остановки мотора.
     Действует при запуске мотора.
     На текущее вращение мотора влияния не оказывает.
     Варианты:
     1) Остановка без отключения подачи электроэнергии,
        средний тормозной путь
        saBrake,'brake','тормоз','тормози','тормозить',1
     2) Остановка с удержанием позиции, минимальный тормозной путь
        saHold,'hold','стоп','стоять','стой',2
     0) Движение накатом, с отключением подачи электроэнергии,
        максимальный тормозной путь
        saCoast,0,'накат' и другие варианты, не входящие в 1) и 2)
    }
    function setHowStop(sa:variant):TEv3TachoMotor;

    {
     Возвращает способ остановки мотора
      1 - Остановка без отключения подачи электроэнергии,
          средний тормозной путь
      2 - Остановка с удержанием позиции, минимальный тормозной путь
      0 - Движение накатом, с отключением подачи электроэнергии,
          максимальный тормозной путь
    }
    function howStop(out sa:variant):TEv3TachoMotor;
    function howStop:variant;

    {
      Устанавливает и возвращает текущую позицию мотора
    }
    function setPosition(p:variant):TEv3TachoMotor;
    function position(out p:variant):TEv3TachoMotor;
    function position:variant;

    {
      Устанавливает скорость вращения от -100% до 100%
      Действует при запуске мотора.
      На текущее вращение мотора влияния не оказывает.
    }
    function setSpeed(sp:variant):TEv3TachoMotor;
    {
      Возвращает установленную ранее скорость вращения от -100 до 100
    }
    function Speed(out sp:variant):TEv3TachoMotor;
    function Speed:variant;

    {
      Возвращает реальную скорость вращения от 0 до 100
    }
    function currentSpeed(out sp:variant):TEv3TachoMotor;
    function currentSpeed:variant;
```

```
    {
      Устанавливает реверс
      Действует при запуске мотора.
      На текущее вращение мотора влияния не оказывает.
    }
    function setReverse(yes:variant):TEv3TachoMotor;
    {
      Возвращает 1, если установлен реверс и 0 в противном случае
    }
    function Reverse(out yes:variant):TEv3TachoMotor;
    function Reverse:variant;

    {
      Останавливает мотор, способ остановки - см. setHowStop
    }
    function Stop(sa:variant):TEv3TachoMotor;
    function Stop():TEv3TachoMotor;

    {
      Запускает мотор (асинхронный вызов). Возможные параметры -
      скорость и способ остановки по умолчанию при вызове stop()
    }
    function Run(sp:variant; sa:variant):TEv3TachoMotor;
    function Run(sp:variant):TEv3TachoMotor;
    function Run():TEv3TachoMotor;

    {
      Запускает мотор на количество миллисекунд (асинхронный вызов).
      Возможные параметры - скорость и способ остановки
    }
    function RunTime(ms:variant; sp:variant; sa:variant):TEv3TachoMotor;
    function RunTime(ms:variant; sp:variant):TEv3TachoMotor;
    function RunTime(ms:variant):TEv3TachoMotor;

    {
      Запускает мотор на количество градусов (асинхронный вызов).
      Возможные параметры - скорость и способ остановки
    }
    function RunDeg(deg:variant; sp:variant; sa:variant):TEv3TachoMotor;
    function RunDeg(deg:variant; sp:variant):TEv3TachoMotor;
    function RunDeg(deg:variant):TEv3TachoMotor;

    {
      Запускает мотор на количество оборотов (асинхронный вызов).
      Возможные параметры - скорость и способ остановки
    }
    function RunTurn(turn:variant; sp:variant; sa:variant):TEv3TachoMotor;
    function RunTurn(turn:variant; sp:variant):TEv3TachoMotor;
    function RunTurn(turn:variant):TEv3TachoMotor;
end;

TEv3LargeMotor=class(TEv3TachoMotor)
 constructor create(port:variant);
 constructor create();
end;

TEv3MediumMotor=class(TEv3TachoMotor)
 constructor create(port:variant);
 constructor create();
end;
```

При создании объектов-моторов можно указывать номер порта.
Действительны следующие значения (регистр символов неважен)
для порта A: 5,'5','outA','A';
для порта B: 6,'6','outB','B';
для порта C: 7,'7','outC','C';
для порта D: 8,'8','outD','D';

Если порт не указывается, то объект-мотор присоединяется к ближайшему (слева-направо, от A до D) минимально нагруженному порту, на котором подключен мотор соответствующего типа. Под нагрузкой порта понимается количество присоединенных к порту объектов. В текущей версии библиотеки не имеет смысла нагружать порт более чем одним объектом.

Примеры:

```
{
 Демонстрация запуска и остановки
 больших моторов (mot1.pp)
 При запуске с блока вместо клавиши Enter
 можно нажимать кнопку CENTER
}
{$mode objfpc}
uses uev3,rubiroMotors;
 var M1,M2:TEv3LargeMotor;
begin
 ev3init();

 // Подключение к первому мотору
 M1:=TEv3LargeMotor.create();
 // запуск на максимальной скорости
 M1.Run(100);
 // единственная возможность остановить мотор - enter
 readln;
 // остановка с торможением
 M1.stop(saBrake);
 readln;

 // Подключение ко второму мотору
 M2:=TEv3LargeMotor.create();
 // запуск на 3 секунды
 M2.RunTime(3000,100,saBrake);
 // можно остановить мотор до завершения вращения
 readln;
 // остановка с заменой торможения на удержание
 M2.stop(saHold);
 readln;

 // запуск моторов в обратную сторону на 5 оборотов
 // с ожиданием завершения их работы
 M1.runTurn(5,-100,saBrake);
 M2.runTurn(5,-100,saBrake).wait();
 writeln ('Motors stopped');
 readln;
end.
```

Модуль rubiroMotors, рулевое управление

Класс **TEv3Rule** используется для создания рулевого управления на двух моторах. В текущей версии библиотеки поддерживается рулевое управление на больших моторах, однако в дорожной карте развития библиотеки есть планы поддержки средних моторов.

```
TEv3Rule=class

  // ждет остановки моторов
  function wait():TEv3Rule;

  constructor create(LPort,RPort:variant);
  constructor create();
  destructor destroy;override;

  // Устанавливает способ остановки моторов (см.TEv3TachoMotor)
  function setHowStop(sa:variant):TEv3Rule;
  // Возвращает способ остановки моторов (см.TEv3TachoMotor)
  function howStop(out sa:variant):TEv3Rule;
  function howStop:variant;

  {
    Устанавливает скорость вращения от -100% до 100% (см.TEv3TachoMotor)
  }
  function setSpeed(sp:variant):TEv3Rule;
  {
    Возвращает установленную ранее скорость вращения от -100 до 100
    (см.TEv3TachoMotor)
  }
  function Speed(out sp:variant):TEv3Rule;
  function Speed:variant;

  {
    Возвращает установленную ранее скорость вращения для левого колеса
  }
  function SpeedLeft(out sp:variant):TEv3Rule;
  function SpeedLeft:variant;

  {
    Возвращает установленную ранее скорость вращения для правого колеса
  }
  function SpeedRight(out sp:variant):TEv3Rule;
  function SpeedRight:variant;

  // устанавливает и возвращает реверс (см.TEv3TachoMotor)
  function setReverse(yes:variant):TEv3Rule;
  function Reverse(out yes:variant):TEv3Rule;
  function Reverse:variant;

  // останавливает оба мотора (см.TEv3TachoMotor)
  function Stop(sa:variant):TEv3Rule;
  function Stop():TEv3Rule;

  {
   запускает оба мотора (асинхронный вызов) (см.TEv3TachoMotor)
   direct - направление движения
   Примеры поведения рулевого управления при положительной скорости
   и значения direct:
    100 - поворот вправо вокруг оси,
    -100 - поворот влево вокруг оси,
    50 - поворот вправо вокруг правого колеса,
    -50 - поворот влево вокруг левого колеса
  }
```

```
    function Run(direct:variant; sp:variant; sa:variant):TEv3Rule;
    function Run(direct:variant; sp:variant):TEv3Rule;
    function Run(direct:variant):TEv3Rule;

    {
      Запускает моторы на количество миллисекунд (асинхронный вызов).
      Возможные параметры - скорость и способ остановки
    }
    function RunTime(direct:variant; ms:variant; sp:variant;
      sa:variant):TEv3Rule;
    function RunTime(direct:variant; ms:variant; sp:variant):TEv3Rule;
    function RunTime(direct:variant; ms:variant):TEv3Rule;

    {
      Запускает моторы на количество градусов (БЛОКИРУЮЩИЙ вызов).
      Возможные параметры - скорость и способ остановки
    }
    function RunDegWait(direct:variant; deg:variant; sp:variant;
      sa:variant):TEv3Rule;
    function RunDegWait(direct:variant; deg:variant; sp:variant):TEv3Rule;
    function RunDegWait(direct:variant; deg:variant):TEv3Rule;

    {
      Запускает моторы на количество оборотов (БЛОКИРУЮЩИЙ вызов).
      Возможные параметры - скорость и способ остановки
    }
    function RunTurnWait(direct:variant; turn:variant; sp:variant;
      sa:variant):TEv3Rule;
    function RunTurnWait(direct:variant; turn:variant; sp:variant):TEv3Rule;
    function RunTurnWait(direct:variant; turn:variant):TEv3Rule;
  end;
```

При создании объекта-рулевого управления можно указывать номера портов для левого и правого больших моторов (идентификация портов - см. выше). Если порты не указывать, будут задействованы два первых свободных больших мотора.

Примеры:

```
{
 Движение по линии до перекрестка
 с помощью двух датчиков света (mot2.pp)
}
{$mode objfpc}
uses uev3,rubiroMotors,rubiroSensors;
var
 L,R:TEv3ColorSensor;
 Rule:TEv3Rule;
{
 Процедур движения по линии до перекрестка
 с помощью двух датчиков цвета

 maxSpeed - максимально развиваемая скорость
 minSpeed - минимальная скорость
 koef - коэффициент поворота, от 0 (поворот отсутсвует)
      до 1 (крайне резкий поворот)
 borderBreak - граница черного, выход из процедуры
      при сумме значений датчиков, меньшей borderBreak
      (выход по перекрестку)
 borderAcc - граница точности, движение прямо при различии
      значений датчиков меньшем borderAcc

}
```

```
procedure run(maxSpeed, minSpeed, koef:double; borderBreak, borderAcc:integer);
 var lref,rref,speed:double;
     direct,prevDirect:double;
begin
 try
  speed:=maxSpeed;
  direct:=0;
  prevDirect:=0;
  Rule.Run(direct,speed);
  while true do begin
    lref:=l.reflect; rref:=r.reflect;
    if lref+rref<borderBreak then exit;
    if abs(lref-rref)<borderAcc then direct:=0
    else direct:=(lref-rref)*koef;
    if (direct=prevDirect)and(speed>=maxSpeed) then continue;
    if (direct=prevDirect) then speed+=1
    else speed:=minSpeed;
    prevDirect:=direct;
    rule.Run(direct,speed);
  end;
 finally
  rule.stop(saBrake);
 end;
end;

begin
 ev3init();
 Rule:=Tev3Rule.Create;
 l:=TEv3ColorSensor.Create;
 r:=TEv3ColorSensor.Create;
 run(100,40,0.45,50,20);
 readln;
end.
```