Справочник по командам EV3 Бейсик

Перевод: Андрей Степанов, «Карандаш и Самоделкин»

Assert

Тестовая функция утверждения помогает проверить на корректность твой код. Добавив утверждение ты сможешь проще найти ошибки в программе. Например, когда часть программы зависит от переменной A, имеющей положительное значение, ты можешь вызвать Assert.Greater(A,0,"A должна быть > 0!"). В случае, если в этом месте программы переменная A будет меньше нуля, выведется указанное тобой предупреждение.

Assert.Failed (сообщение)

Вывести предупреждение на экран. Используй эту функцию только если наблюдаешь ошибки в программе и тебе нужно их найти.
Сообщение: Сообщение, которое будет выведено на экран

Assert.Equal(A, B, сообщение)

Утверждаем, что два значения равны. проверяется полное совпадение, даже "True" и "tRue" считаются различными выражениями.
A: Первый параметр
B: Второй параметр
Сообщение: Сообщение, которое будет выведено, если утверждение ошибочно

Assert.NotEqual(A, B, сообщение)

Утверждаем, что два значения не равны. проверяется полное совпадение, даже "True" и "tRue" считаются различными выражениями.
A: Первый параметр
B: Второй параметр
Сообщение: Сообщение, которое будет выведено, если утверждение ошибочно

Assert.Less(A, B, сообщение)

Утверждаем, что первое число меньше второго.
A: Первый параметр
B: Второй параметр
Сообщение: Сообщение, которое будет выведено, если утверждение ошибочно

Assert.Greater(A, B, сообщение)

Утверждаем, что первое число больше второго.
A: Первый параметр
B: Второй параметр
Сообщение: Сообщение, которое будет выведено, если утверждение ошибочно

Assert.LessEqual(A, B, сообщение)

Утверждаем, что первое число меньше или равно второму.
A: Первый параметр
B: Второй параметр
Сообщение: Сообщение, которое будет выведено, если утверждение ошибочно

Assert.GreaterEqual(A, B, сообщение)

Утверждаем, что первое число больше или равно второму.
A: Первый параметр
B: Второй параметр

Сообщение: Сообщение, которое будет выведено, если утверждение ошибочно

Assert.Near(A, B, сообщение)
Утверждаем, что числа почти равны. Это может быть полезно при сравнении двух дробных чисел с большим количеством знаков после запятой, а их округление может вызвать немного различные результаты
A: Первый параметр
B: Второй параметр
Сообщение: Сообщение, которое будет выведено, если утверждение ошибочно

Buttons

Прочитать состояния и нажатия (включая щелчки) кнопок на блоке EV3. Кнопки обозначаются с использованием следующих символов:

U Вверх
D Вниз
L Влево
R Вправо
E Центр

Buttons.GetClicks()
Проверяет какие кнопки были нажаты с момент последнего вызова GetClicks и возвращает текст, содержащий соответствующие им символы. Состояние кнопок при этом очищается. Также издает звук, когда зафиксировано нажатие кнопок.
Возвращает: Текст, содержащий символы нажатых кнопок (может быть пустым, если ничего не нажато)

Buttons.Wait ()
Ждать, пока одна из кнопок на блоке не будет нажата.

Buttons.Flush ()
Очистить состояние нажатых кнопок. Последующие вызовы GetClicks покажет только те кнопки, которые были нажаты после очистки.

Buttons.Current
Кнопки, которые в данный момент нажаты. Содержит текст с символами, соответствующим всем кнопкам, нажатым в данный момент.

EV3File

Доступ к файловой системе блока EV3 для чтения или записи данных. К именам файлов домет быть добавлен абсолютный путь начиная с '/' для доступа к любому файлу в файловой системе или относительный, начиная с папки prjs.

EV3File.OpenWrite(имя_файла)
Открывает файл для записи. если файл уже существует, он будет перезаписан.
имя_файла: Имя файла для записи/перезаписи
Возвращает: Номер, идентифицирующий данный открытый файл (file handle)

EV3File.OpenAppend(имя_файла)
Открывает файл для добавления в него данных. Если файл не существует, он будет создан.
имя_файла: Имя файла для создания/добавления данных
Возвращает: Номер, идентифицирующий данный открытый файл (file handle)

EV3File.OpenRead(имя_файла)
Открывает файл для чтения. Если файл не существует, возвращает 0.
имя_файла: Имя файла для чтения
Возвращает: Номер, идентифицирующий данный открытый файл (file handle) или 0, если файл не существует

EV3File.Close(идентификатор)
Закрывает открытый файл.
Идентификатор: Идентификатор файла (file handle), полученный в момент открытия файла

EV3File.WriteLine(идентификатор, текст)
Записывает текстовую строку в файл. Строка будет записана в кодировке ISO-8859-1 и оканчиваться символом перевода строки (код 10).
Идентификатор: Идентификатор файла (file handle), полученный в момент открытия файла
Текст: Текст для записи в файл

WriteByte(идентификатор, число)
Записывает один байт данных в файл.
Идентификатор: Идентификатор файла (file handle), полученный в момент открытия файла
Число: Один байт для записи (значение от 0 до 255).

EV3File.ReadLine(идентификатор)
Прочитать текстовую строку из файла. Строка будет прочитана в кодировке ISO-8859-1 и должна оканчиваться символом перевода строки (код 10).
Идентификатор: Идентификатор файла (file handle), полученный в момент открытия файла
Возвращает: Текстовая строка, прочитанная из текущей строки файла

EV3File.ReadByte(идентификатор)
Прочитать один байт из файла.
Идентификатор: Идентификатор файла (file handle), полученный в момент открытия файла
Возвращает: Следующий по счету байт из файла (число)

EV3File.ReadNumberArray(идентификатор, размер)
Читает массив чисел из бинарного файла. Числа будт раскодированы с использованием кодировки IEEE как тип Float с одинарной точностью
Идентификатор: Идентификатор файла (file handle), полученный в момент открытия файла
Размер: Размер массива, который будет прочитан
Возвращает: Массив чисел указанного размера

EV3File.ConvertToNumber(текст)
Конвертация текста в число.
Текст: Текст, содержащий числа, может содержать также дробную часть числа
Возвращает: Число

EV3File.TableLookup(имя_файла, байт_в_строке, строка, столбец)
Функция предназначена для чтения байт из потенциально огромных файлов, которые слишком велики, чтобы быть переданы в память в целом. Поскольку файл может быть настолько большой, что численной адресации будет недостаточно, вводятся параметры строка/столбец
имя_файла: Имя файла
байт_в_строке: Если файл имеет структуру строк/столбцов, в параметре указывается число байт в строке, иначе указывай 1
строка: Какую строку читать (начиная с 0)
столбец: Какой столбец читать (начиная с 0)
Возвращает: Байт из указанной позиции

LCD
Управление ЖК-экраном на блоке EV3. EV3 имеет черно-белый дисплей с разрешением 178x128 точек. Координаты отсчитываются от левого верхнего угла экрана, имеющего координаты 0,0

LCD.StopUpdate()
Начинать запоминать всю выводимую экран информацию без ее отображения. При следующем вызове Update() все накопленные изменения наконец-то отобразятся. Вы можете использовать эту функцию, чтобы предотвратить мерцание или для ускорения рисования сложных изображений в ЖК-дисплее.

LCD.Update()
Вывести на экран все изменения, которые произошли с момента последнего вызова StopUpdate().

LCD.Clear()
Очистка экрана. Все пиксели закрашиваются в белый цвет.

LCD.Pixel(цвет, x, y)
Вывод на экран одиночного пикселя указанного цвета.
Цвет: 0 (белый) или 1 (черный)
X: Координата X по горизонтали
Y: Координата Y по вертикали

LCD.Line (цвет, x1, y1, x2, y2)
Вывод на экран линии указанного цвета.
Цвет: 0 (белый) или 1 (черный)
x1: Координата X начальной точки
y1: Координата Y начальной точки
x2: Координата X конечной точки
y2: Координата Y конечной точки

LCD.Circle (цвет, x, y, радиус)
Рисует не закрашенный круг указанного размера.
Цвет: 0 (белый) или 1 (черный)
X: Координата X центра круга
Y: Координата Y центра круга
Радиус: Радиус круга

LCD.Text (цвет, x, y, размер_шрифта, текст)
Напечатать заданный текст или числа на экране.
Цвет: 0 (белый) или 1 (черный)
X: Координата X начала печати
Y: Координата Y начала печати
Размер_шрифта: Размер шрифта: 0 (маленький), 1 (средний), 2 (большой)
Текст: Текст или числа для вывода на дисплей

LCD.Write (x, y, текст)
Напечатать заданный текст черным цветом на экране. Если нужно выводить текст с выбором цвета и размера, используй LCD.Text
X: Координата X начала печати
Y: Координата Y начала печати
Текст: Текст или числа для вывода на дисплей

LCD.FillRect (цвет, x, y, ширина, высота)
Рисует закрашенный заданным цветом прямоугольник.
Цвет: 0 (белый) или 1 (черный)
X: Координата X левой стороны прямоугольника
Y: Координата Y верхней стороны прямоугольника
Ширина: Ширина прямоугольника
Высота: Высота прямоугольника

LCD.Rect (цвет, x, y, ширина, высота)
Рисует не закрашенный прямоугольник с границей заданного цвета.
Цвет: 0 (белый) или 1 (черный)
X: Координата X левой стороны прямоугольника
Y: Координата Y верхней стороны прямоугольника
Ширина: Ширина прямоугольника
Высота: Высота прямоугольника

LCD. InverseRect (цвет, x, y, ширина, высота)
Инвертирует все пиксели в границах прямоугольника.
Цвет: 0 (белый) или 1 (черный)
X: Координата X левой стороны прямоугольника
Y: Координата Y верхней стороны прямоугольника
Ширина: Ширина прямоугольника
Высота: Высота прямоугольника

LCD. FillCircle(цвет, x, y, радиус)
Рисует закрашенный круг указанного размера.
Цвет: 0 (белый) или 1 (черный)
X: Координата X центра круга
Y: Координата Y центра круга
Радиус: Радиус круга

LCD.BmpFile (цвет, x, y, имя_файла)
Вывести файл заданным цветом на экран. Поддерживаются только файлы в формате .rgf
Цвет: 0 (белый) или 1 (черный)
X: Координата X левой стороны изображения
Y: Координата Y верхней стороны изображения
Имя_файла: имя файла без расширения .rgf. Может содержать относительный путь от папки 'prjs' или абсолютный, начинаясь с '/').

EV3

Полезные функции блока EV3.

EV3.SetLEDColor (цвет, эффект)
Установить цвет подсветки кнопок блока EV3 и ее эффекты. Подсветка: OFF - выключена, GREEN - зеленая, RED - красная, ORANGE - оранжевая. Эффекты: NORMAL - гореть постоянно, FLASH - вспыхивать, PULSE - пульсировать.
Цвет: Может быть "OFF", "GREEN", "RED", "ORANGE"
Эффект: Может быть "NORMAL", "FLASH", "PULSE"

EV3.SystemCall (системная_команда)
Запустить одну системную команду в операционной системе EV3 Linux. Выполняемая программа будет ждать, пока запущенная в операционной системе программа не завершит работу.
системная_команда: Системная команда. Статус завершения запущенной команды

EV3.QueueNextCommand()
Повышение производительности при работе в режиме "ПК". Не посылать следующую команду на блок сразу же, а подождать, поставив ее в очередь. В режиме запуска программы на блоке не имеет смысла.

EV3.Time
Время в миллисекундах, прошедшее с момента запуска программы.

EV3.BatteryLevel
Возвращает текущий заряд батарей, от 0 до 100.

Motor

Управление моторами, подлеченными к блоку EV3. Для функции Motor необходимо указать один или несколько портов, к которым подключены моторы, например "A", "BC", "ABD". Для блоков, подключенных к главному блоку по USB порты указываются так: 3BC", "2A". В этом режиме только двигатели одного блока доступны в одной команде. Скорость и Мощность - разные понятия. При вызове команды для вращения мотора с постоянной скоростью, электрическая мощность, подаваемая на него будет автоматически регулироваться, чтобы поддерживать заданную скорость. При вызове команды для вращения мотора с постоянной мощностью, его скорость будет зависеть от сопротивления, которое он испытывает во время работы.

Motor.Stop (порты, торможение)
Остановить один или несколько моторов. Команда завершает так же все запланированные или незавершенные команды управления этими моторами.

Порты: Порт(ы) моторов
Торможение: "True", если необходимо удерживать положение после останови моторов

Motor.Start (порты, скорость)
Запустить один или несколько моторов с указанной скоростью или изменить скорость уже запущенных моторов на указанную.
Порты: Порт(ы) моторов
Скорость: Скорость от -100 (полный назад) до 100 (полный вперед).

Motor.StartPower (порты, мощность)
Запустить один или несколько моторов с указанной мощностью или изменить мощность уже запущенных моторов на указанную.
Порты: Порт(ы) моторов
Мощность: Мощность от -100 (полный назад) до 100 (полный вперед).

Motor.StartSync (порты, скорость1, скорость2)
Синхронно запустить два мотора с указанными скоростями в режиме контроля за их вращением. Если один мотор будет испытывать нагрузку, которая замедлит его скорость, второй мотор пропорционально замедлится, чтобы сохранить траекторию движения.
Порты: Имена двух портов для моторов (например "AB" или "CD").
Скорсоть1: Скорость от -100 (полный назад) до 100 (полный вперед) мотора с младшим по алфавиту номером порта
Скорость2: Скорость от -100 (полный назад) до 100 (полный вперед) мотора со старшим по алфавиту номером порта

Motor.GetSpeed (порт)
Запросить текущую скорость мотора
Порт: Порт мотора
Возвращает: Текущая скорость в диапазоне от -100 до 100

Motor.IsBusy (порты)
Проверить один или несколько моторов на занятость.
Порты: Порты моторов
Возвращает: "True" если хотя бы один из моторов запущен, "False" в противном случае.

Motor.Schedule (порты, скорость, зона1, зона2, зона3, торможение)
Запустить один или несколько моторов с указанной скоростью с возможностью плавного старта и плавного замедления. Скорость может регулироваться в пределах общего угла поворота, в зоне1 двигатель разгоняется до указанной скорости, в зоне 2 поддерживает ее, в зоне 3 - замедляется до полной остановки. Полный угол, на который повернется мотор - это сумма углов поворота зона1 + зона2 + зона3. Команда передает управление в программу сразу, не ждет, пока мотор повернется. Чтобы отследить окончания отработки команды, используйте IsBusy(). Чтобы дождаться конца, пока мотор закончит отработку команды - используйте Wait().
Порты: Порты моторов
Скорость: Скорость от -100 (полный назад) до 100 (полный вперед)
зона1: часть общего угла поворота мотора, в пределах которой мотор разгоняется
зона2: часть общего угла поворота мотора, в пределах которой мотор поддерживает указанную скорость
зона3: часть общего угла поворота мотора, в пределах которой мотор плавно замедляется до полной остановки
Возвращает: "True", если необходимо удерживать положение после останови моторов

Motor.SchedulePower (порты, мощность, зона1, зона2, зона3, торможение)

Запустить один или несколько моторов с указанной мощностью с возможностью плавного старта и плавного замедления. Мощность может регулироваться в пределах общего угла поворота, в зоне1 двигатель плавно наращивает мощность до указанной, в зоне 2 поддерживает ее, в зоне 3 - плавно уменьшает подаваемую мощность до полной остановки. Полный угол, на который повернется мотор - это сумма углов поворота зона1 + зона2 + зона3. Команда передает управление в программу сразу, не ждет, пока мотор повернется. Чтобы отследить окончания отработки команды, используйте IsBusy(). Чтобы дождаться конца, пока мотор закончит отработку команды - используйте Wait().
Порты: Порты моторов
Мощность: Мощность от -100 (полный назад) до 100 (полный вперед)
зона1: часть общего угла поворота мотора, в пределах которой мотор разгоняется
зона2: часть общего угла поворота мотора, в пределах которой мотор поддерживает указанную скорость
зона3: часть общего угла поворота мотора, в пределах которой мотор плавно замедляется до полной остановки
Возвращает: "True", если необходимо удерживать положение после останови моторов

Motor.ScheduleSync (порты, скорость1, скорость2, угол, торможение)

Поворачивает два мотора синхронно на определенное количество градусов. Синхронная работа двигателей означает, что когда один двигатель нагружен и что-то препятствует его вращению, второй двигатель пропорционально замедлится или даже вообще остановится. Это команда особенно полезна для колесных роботов - для возможности сохранить траекторию движения. Угол, на который будет повернуты моторы относится к мотору с наибольшей скоростью вращения, угол поворота второго мотора будет рассчитан пропорционально его скорости. Команда передает управление в программу сразу, не ждет, пока мотор повернется. Чтобы отследить окончания отработки команды, используйте IsBusy(). Чтобы дождаться конца, пока мотор закончит отработку команды - испольузйте Wait().
Порты: Имена двух портов для моторов (например "AB" или "CD")
Скорость1: Скорость от -100 (полный назад) до 100 (полный вперед) мотора с младшим по алфавиту номером порта
Скорость2: Скорость от -100 (полный назад) до 100 (полный вперед) мотора со старшим по алфавиту номером порта
Угол: Угол поворота (мотора с наибольшей скоростью)
Возвращает: "True", если необходимо удерживать положение после останови моторов

Motor.ResetCount (порты)

Сбросить счетчик оборотов одного или несколько моторов в 0.
Порты: Порты моторов

Motor.GetCount (порт)

Запросить данные счетчика оборота с указанного мотора. Счетчик оборотов работает даже если мотор вращается не по командам блока EV3, а путем приложения внешней силы.
Порт: Порт мотора
Возвращает: Текущее значение счетчика оборотов мотора в градусах

Motor.Move (порты, скорость, угол, торможение)

Поворачивает один или несколько моторов с заданной скоростью на указанный угол (в градусах). Программа не будет переходить к выполнению следующих команд до тех пор, пока моторы не повернутся на требуемый угол. Если тебе нужно больше возможностей по

управлению моторами, например плавный старт и замедление, используй команду Motor.Schedule
Порты: Порты моторов
Скорость: Скорость от -100 (полный назад) до 100 (полный вперед)
Угол: Угол поворота
Возвращает: "True", если необходимо удерживать положение после останови моторов

Motor.MovePower (порты, скорость, угол, торможение)
Поворачивает один или несколько моторов с заданной мощностью на указанный угол (в градусах). Программа не будет переходить к выполнению следующих команд до тех пор, пока моторы не повернутся на требуемый угол. Если тебе нужно больше возможностей по управлению моторами, например плавное наращивание мощности или ее уменьшение, используй команду Motor.SchedulePower
Порты: Порты моторов
Мощность: Мощность от -100 (полный назад) до 100 (полный вперед)
Угол: Угол поворота
Возвращает: "True", если необходимо удерживать положение после останови моторов

Motor.MoveSync (порты, скорость1, скорость2, угол, торможение)
Поворачивает два мотора синхронно на определенное количество градусов. Синхронная работа двигателей означает, что когда один двигатель нагружен и что-то препятствует его вращению, второй двигатель пропорционально замедлится или даже вообще остановится. Это команда особенно полезна для колесных роботов - для возможности сохранить траекторию движения. Угол, на который будет повернуты моторы относится к мотору с наибольшей скоростью вращения, угол поворота второго мотора будет рассчитан пропорционально его скорости.
Порты: Имена двух портов для моторов (например "AB" или "CD")
Скорость1: Скорость от -100 (полный назад) до 100 (полный вперед) мотора с младшим по алфавиту номером порта
Скорость2: Скорость от -100 (полный назад) до 100 (полный вперед) мотора со старшим по алфавиту номером порта
Угол: Угол поворота (мотора с наибольшей скоростью)
Возвращает: "True", если необходимо удерживать положение после останови моторов

Motor.Wait (порты)
Подождать, пока моторы не закончат выполнять команды Schedule или Move. Использование этой команды, как правило, лучше, чем вызов IsBusy() в цикле.
Порты: Порты моторов

Sensor
Работа с датчиками, подключенными к блоку EV3. Чтобы указать локальный датчик, используется номер порта 1..4, для указания номеров портов на первом в цепочке подключенных по USB блоках используйте 5..8, на третьем - 9..12

Sensor.GetName (порт)
Запросить имя и текущий режим работы датчика, подключенного к указанному порту. Эта команда предназначена прежде всего для диагностики, т.к. ты как правило знаешь, куда и какие датчики подключил
Порт: Номер порта датчика
Возвращает: Текст описания (например "TOUCH")

Sensor.GetType (порт)
Запросить цифровой идентификатор датчика, подключенного к указанному порту.
Порт: Номер порта датчика
Возвращает: Цифровой идентификатор датчика (например 16 для датчика нажатия)

Sensor.GetMode (порт)
Запросить текущий режим работы датчика, подключенного к указанному порту. Многие датчики умеют работать в различных режимах. Например цветосветовой датчик умеет работать с режиме измерение отраженного света, в режиме измерения внешней освещенности, в режиме измерения цвета. При подключении датчика он как правило работает в режиме 0, затем режим можно изменить в программе.
Порт: Номер порта датчика
Возвращает: Текущий режим работы датчика (0 для режима по умолчанию)

Sensor.SetMode (порт, режим)
Переключить режим работы датчика, подключенного к указанному порту. Датчики могут работать с различных режимах, например: EV3 датчик цвета, режимы 0 - отраженный свет, 1 - внешняя освещенность, 2 - цвет. Для уточнения какой датчик в какой режим можно переключить - смотрите документацию. Программа не продолжится до тех пор, пока датчик не переключится в новый режим и данные с него не будут доступны. Обратите внимание, что датчик будет оставаться в выбранном режиме даже после того, как программа завершит работу, поэтому чтобы избежать путаницы всегда устанавливайте режим в начале программы перед использованием датчика.
Порт: Номер порта датчика
Режим: Новый режим работы из перечня разрешенных для данного типа датчика

Sensor.IsBusy (порт)
Проверить, не занят ли датчик в данный момент сменой режима или инициализацией. Во время переключения датчик некоторое время не доступен (занят).
Порт: Номер порта датчика
Возращение: "True" если датчик занят (не доступен)

Sensor.Wait (порт)
Дождаться, пока датчик не переключится в новый режим или не инициализируется. Обычно тебе не нужно использовать эту команду, так как SetMode() дожидается смены режима. Если ты используешь параллельные задачи - без этой команды не обойтись - в ней датчик мог менять режим и стать на время недоступным.
Порт: Номер порта датчика

Sensor.ReadPercent (порт)
Прочитать текущее показание датчика с отображением его в процентах (0..100). Большинство датчиков могут перевести свои показания в проценты, например у датчика нажатия 0 - не нажат, 100 - нажат.
Порт: Номер порта датчика
Возвращает: Текущее показание датчика
(Пример для датчика освещенности 0 - очень темно, 100 - очень светло)

Sensor.ReadRaw (порт, сколько_значений)

Прочитать текущее показание датчика в "сыром" режиме, без перевода в проценты. В некоторых режимах показания датчика не могут быть переведены в проценты, например цвета, в этом случае используется команда чтения ReadRaw.
Порт: Номер порта датчика
Сколько_значений: Размер возвращаемого массива с показаниями
Возвращает: Массив с показаниями датчика с требуемым числом элементов. Массив начинается с 0 элемента. Элементы, в которых данные не получены, будут равны 0

Sensor.ReadRawValue (порт, элемент)
Похожа на ReadRaw, но возвращает только одно "сырое" показание с датчика, а не массив показаний. Например для работы с цветосветовым датчиком в режиме определения цвета используйте функцию ReadRawValue(порт, 0).
Порт: Номер порта датчика
Элемент: Индекс элемента массива с показаниями, начиная с 0.
Возвращает: Один элемент массива с показаниями

Sensor.CommunicateI2C (порт, адрес, writebytes, readbytes, writedata)
Взаимодействие с устройством, работающим по протоколу I2C, подключенному к одному из портов датчиков. Эта команда может посылать и принимать несколько байт по шине I2C, с ее помошью можно работать с датчиками сторонних производителей, использующих этот протокол обмена или, например, с Arduino. Обратите внимание, команда работает в пределах одного блока EV3 и только с подчиненными (slave) I2C устройствами.
Порт: Номер порта датчика
Адрес (0 - 127) подчиненного I2C устройства на шине I2C
writebytes: Количество байт для отправки I2C-устройству (максимум 31).
readbytes Количество байт для приема с I2C-устройства (максимум 32, минимум 1).
writedata массив, содержащий байты для отправки (индекс первого элемента - 0)
Возвращает: Массив, содержащий запрошенное количество принятых байт с I2C-устройства (индекс первого элемента - 0)

Sensor.SendUARTData (порт, writebytes, writedata)
Посылает данные по протоколу UART устройству, подключенному к порту датчика. Функция полезна для общения с самодельными датчиками и исполнительными устройствами.
Порт: Номер порта датчика
writebytes Количество байт для отправки на устройство (максимум 32).
writedata Массив, содержащий байты для отправки по UART (младший индекс 0).

Bluetooth
Отправка сообщений между блоками EV3 посредством Bluetooth

Mailbox.Create (имя_ящика)
Создать почтовый ящик с указанным именем, который будет принимать входящие сообщения, адресованные этому блоку EV3 в данный ящик. Только после создания ящика входящие сообщения будут сохраняться. Максимальное количество создаваемых ящиков - 30.
Имя_ящика Имя почтового ящика для создания Цифровой идентификатор почтового ящика. Необходим для получения сообщения из ящика

Mailbox.Send (имя_блока_EV3, имя_ящика, сообщение)
Отправить сообщение на другой блок EV3 в указанный почтовый ящик.
имя_блока_EV3 Имя блока EV3, на который отправляется сообщение. Bluetooth-соединение с блоком должно быть предварительно установлено. Это поле можно оставить пустым, тогда сообщение будет отправлено всем подключенным блокам EV3.
имя_ящика Имя почтового ящика, в который будет отправлено сообщение
Сообщение в виде текста. Пока поддерживаются только текстовые сообщения

Mailbox.IsAvailable (идентификатор)
Проверка наличия нового сообщения в указанном локальном почтовом ящике.
Идентификатор: Цифровой идентификатор почтового ящика "True" если сообщение пришло. "False" в противном случае

Mailbox.Receive (идентификатор)
Получить последнее сообщение из указанного локального почтового ящика. Если ящик пуст, программа будет ожидать до тех пор, пока не появится новое сообщение в ящике. После прочтения сообщение будет удалено и следующий вызов Receive будет ожидать нового сообщения в ящике. Чтобы избежать блокировки в его ожидании, используйте IsAvailable(). Если почтовый ящик с таким именем не существует, команда возвратит пустой текст.
Идентификатор: Цифровой идентификатор почтового ящика Сообщение в виде текста. Пока поддерживаются только текстовые сообщения

Mailbox.Connect (имя_блока_EV3)
Установить соединение с другим блоком EV3 по Bluetooth. Только после того как соединение установлено (этой командой или из меню блока EV3) могут отправляться и приниматься сообщения.
Имя_блока_EV3: Имя блока EV3, с которым нужно установить соединение

Speaker
Использование динамика на блоке EV3 для воспроизведения тонов, нот или звуковых файлов.

Speaker.Stop ()
Остановить звучание любого проигрываемого в данный момент тона или звукового файла.

Speaker.Tone (громкость, частота, длительность)
Воспроизвести тон указанной частоты и длительности.
Громкость тона 0 - 100
Частота, Гц в диапазоне 250 - 10000
Длительность тона в миллисекундах

Speaker.Note (громкость, нота, длительность)
Воспроизвести ноту указанной высоты и длительности.
Громкость 0 - 100
Нота в виде текста, от "C4" до "B7" с полутонами в виде "C#5"
Длительность ноты в миллисекундах

Speaker.Play (громкость, имя_файла)

Воспроизвести звуковой файл, хранящийся на блоке. Поддерживаются файлы только в формате .rsf
Громкость 0 - 100
Имя_файла Имя звукового файла без расширения .rsf Имя файла может содержать относительный папки 'prjs' путь иил абсолютный, начиная с корня ФС '/'

Speaker.IsBusy ()
Проверить, занят ли динамик воспроизведением тона/ноты/звука
Возвращает: "True", если звук в данный момент проигрывается, "False" в противном случае.

Speaker.Wait ()
Дождаться, пока проигрываемый в данный момент закончит воспроизводиться. Если звук не воспроизводится, команда сразу же передает управлению программе.

Thread
Этот объект предназначен для организации потоков в программе. Поток представляет собой фрагмент программного кода, который может работать независимо и параллельно основной программе. Например ты можешь создать поток, который будет управлять двигателями в то время как основная программа будет опрашивать датчики или ожидать действий пользователя. Вообще говоря, многопоточность - сложная штука, не для начинающих. В ней стоит разобраться дополнительно.

Thread.Yield ()
Уступить процессор другому потоку. Команда дает знать другим потокам, что он простаивает и они могут более активно занимать ресурсы процессора. Потоки на самом деле не выполняются параллельно, процессор переключается между потоками настолько быстро, что создается иллюзия параллельного их выполнения. Если в потоке выполняется какое-то ожидание или бездействие, с помощью этой команды можно явно это сообщить для повышения скорости работы других потоков.

Thread.CreateMutex ()
Создать мьютекс (сокр. от "mutual exclusion") - взаимное исключение, который может быть использован для синхронизации потоков. Мьютекс может быть только создан, но не удален. Лучше всего создать все необходимые мьютексы при старте программы и держать их номера в глобальных переменных.
Возвращает: Идентификатор мьютекса. Будет использоваться в дальнейшем для блокировки и разблокировки

Thread.Lock (идентификатор)
Попытаться блокировать указанный мьютекс. Установка блокировки не позволит никакому другому потоку установить блокировку на него.
Идентификатор мьютекса (полученный от CreateMutex())

Thread.Unlock (идентификатор)
Снять блокировку указанного мьютекса. Команду можно использовать только в случае, если блокировка действительно была ранее установлена.
Идентификатор мьютекса (полученный от CreateMutex())

Thread.Run

Создать новый поток. Просто назначьте подпрограмму и она начнет выполняться в качестве независимого потока (пример Thread.Run = MYSUB). Любая подпрограмма может быть преобразована в независимы поток, однако одна подпрограмма может быть преобразована в поток только однократно, два потока из нее сделать не удастся. При использовании Thread.Run в то время, как подпрограмма уже работает, вызов потока поставится в очередь до тех пор, пока она не завершит свою работу. Это обеспечит корректную работу, но запуск потока вероятно может произойти позже запланированного. Все запущенные потоки остановятся, как только основная программа завершит свою работу.

Vector

Объект для выполнения операций над большим количеством чисел. Они называются векторами и будут храниться с использованием массивов с последовательными индексами (начиная с 0). В случае выполнений операций над массивами с различным числом элементов, недостающие элементы массивов будут рассматриваться кк элементы, содержащие 0.

Vector.Init (размер, значение)

Создать вектор заданного размера с указанным количеством элементов, содержащих одинаковое число.
Размер вектора
Значение, для всех элементов
Возвращает: Созданный вектор

Vector.Add (размер, A, B)

Сложение двух векторов путем сложения соответствующих элементов (C[0]=A[0]+B[0], C[1]=A[1]+B[1]...)
Размер: Сколько элементов брать складывать
A: Первый вектор
B: Второй вектор Вектор, содержащий результат сложения

Vector.Sort (размер, A)

Сортировать элементы вектора в порядке возрастания.
Размер: Количество элементов для сортировки
A: Массив, содержащий элементы для сортировки
Возвращает: Вектор, содержащий элементы в порядке возрастания значений

Vector.Multiply (строк, столбцов, k, A, B)

Операция умножения матриц. Входные вектора рассматриваются как двумерные матрицы заданной ширины и высоты. Отдельные строки матрицы хранятся в векторе непосредственно одна за другой. Чтобы узнать больше об этой математической операции, см. http://en.wikipedia.org/wiki/Matrix_multiplication
Строк: Количество строк в результирующей матрице
Столбцов: Количество столбцов в результирующей матрице
k: Количество столбцов в исходной матрице A и количество строк в исходной матрице B
A: Матрица A размера строк * k
B: Матрица B размера k * столбцов
Возвращает: Матрица, содержащая результаты умножения