Казанский (Приволжский) федеральный университет
СУНЦ IT-лицей КФУ

Л.Р. Шарафеева
В.С. Гуськов
Е.В. Сазонов

Программирование антропоморфных роботов

Методическая библиотека

КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

**Специализированный учебный научный центр –
общеобразовательная школа-интернат «IT-лицей»**

Л.Р. ШАРАФЕЕВА, В.С. ГУСЬКОВ, Е.В. САЗОНОВ

ПРОГРАММИРОВАНИЕ АНТРОПОМОРФНЫХ РОБОТОВ

Учебное пособие

КАЗАНЬ

2021

УДК 621.865.8
ББК 32.816
Ш25

Печатается по рекомендации педагогического совета
Специализированного учебного научного центра –
общеобразовательная школа-интернат «IT-лицей»
ФГАОУ ВО «Казанский (Приволжский) федеральный университет»
(протокол № 2 от 15.10.2021)

Авторы:
Л.Р. Шарафеева, старший преподаватель кафедры математики
и прикладной информатики Елабужского института КФУ
В.С. Гуськов, заместитель директора по учебной работе СУНЦ IT-лицей КФУ
Е.В. Сазонов, учитель технологии, педагог дополнительного образования
СУНЦ IT-лицей КФУ

Рецензенты:
директор комплексного центра обучения в сфере энергоэффективности
Инженерного института ФГАОУ ВО «Казанский (Приволжский) федеральный университет»,
кандидат технических наук, доцент **Е.С. Каратаева;**
старший преподаватель кафедры математики и прикладной информатики
Елабужского института ФГАОУ ВО «Казанский (Приволжский)
федеральный университет» **Е.М. Любимова**

Шарафеева Л.Р.
Ш25 **Программирование антропоморфных роботов** [Электронный ресурс]: учеб. пособие / Л.Р. Шарафеева, В.С. Гуськов, Е.В. Сазонов. – Электрон. текстовые дан. (1 файл: 1,31 Мб). – Казань: Издательство Казанского университета, 2021. – 76 с. – Систем. требования: Adobe Acrobat Reader. – Режим доступа: https://kpfu.ru/portal/docs/F1313342253/Programmirovanie.antropomorfnykh.robotov.AR_101M.pdf. – Загл. с титул. экрана.

ISBN 978-5-00130-552-1

Учебное пособие посвящено изучению программирования андроидного робота AR-101M. В пособии содержатся сведения об основных элементах управления языка программирования AR-Basic Studio, типах данных, реализации основных алгоритмических структур. Рассмотрено большое количество примеров разработки программ, приведены задания для самостоятельного решения.

Пособие предназначено для школьников 8–10 классов, желающих познакомиться с основами программирования андроидных роботов серии AR- 101M.

УДК 621.865.8
ББК 32.816

ISBN 978-5-00130-552-1

СОДЕРЖАНИЕ

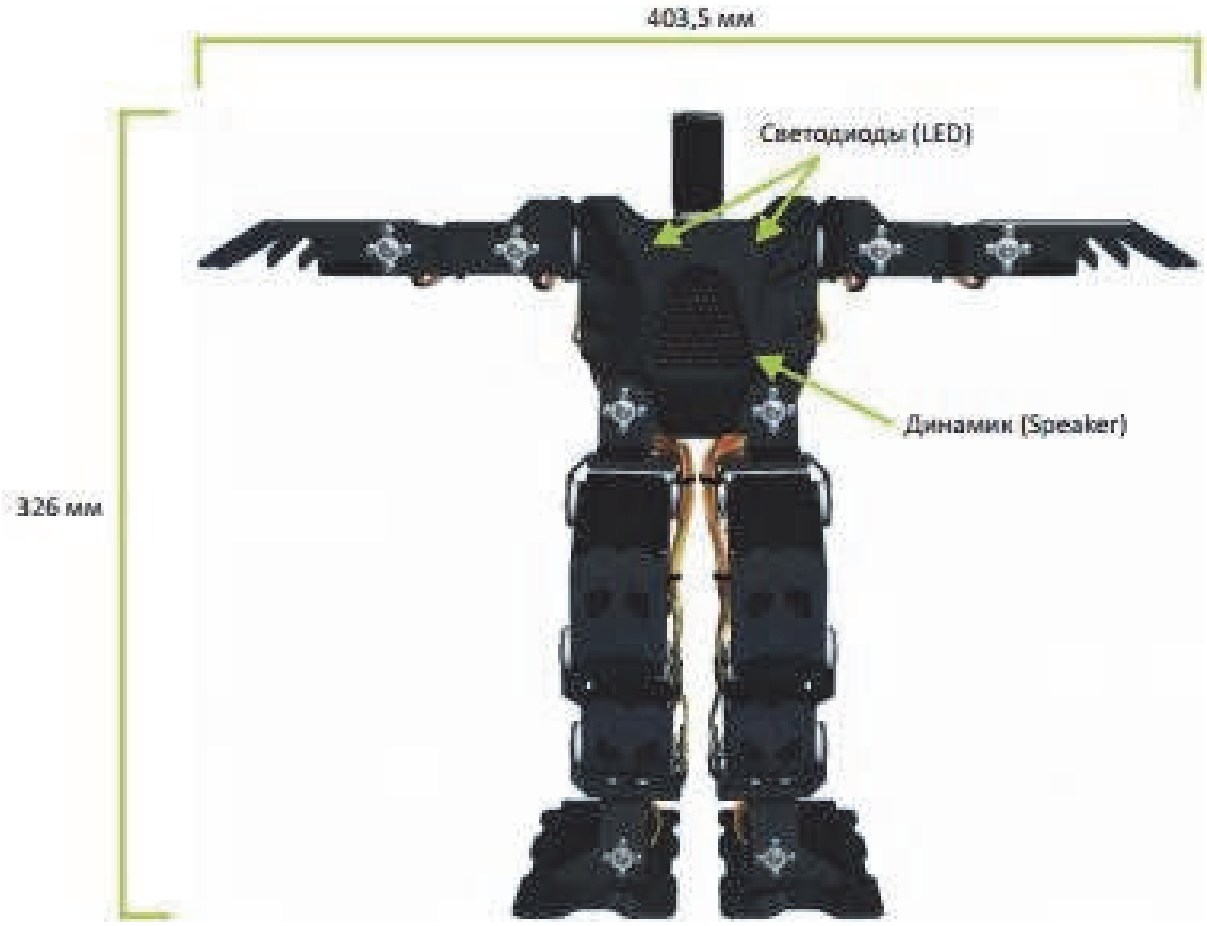
Урок 1. Знакомство с антропоморфным роботом AR-101M

План

1. Знакомство с антропоморфным роботом AR-101M. Использование дополнительного оборудования: гироскоп, акселерометр, датчики.
2. Демонстрация учителем возможностей андроидного робота AR-101M.
3. Домашнее задание.

Описание антропоморфного робота AR-101M

Антропоморфный (человекоподобный) робот AR-101M (рис. 1) – это высокотехнологичное электромеханическое изделие, правдоподобно имитирующее основные движения человеческого тела. Включая подлинное (с отрывом стопы от поверхности) прямохождение, спортивную, танцевальную деятельность и ряд иных возможностей.



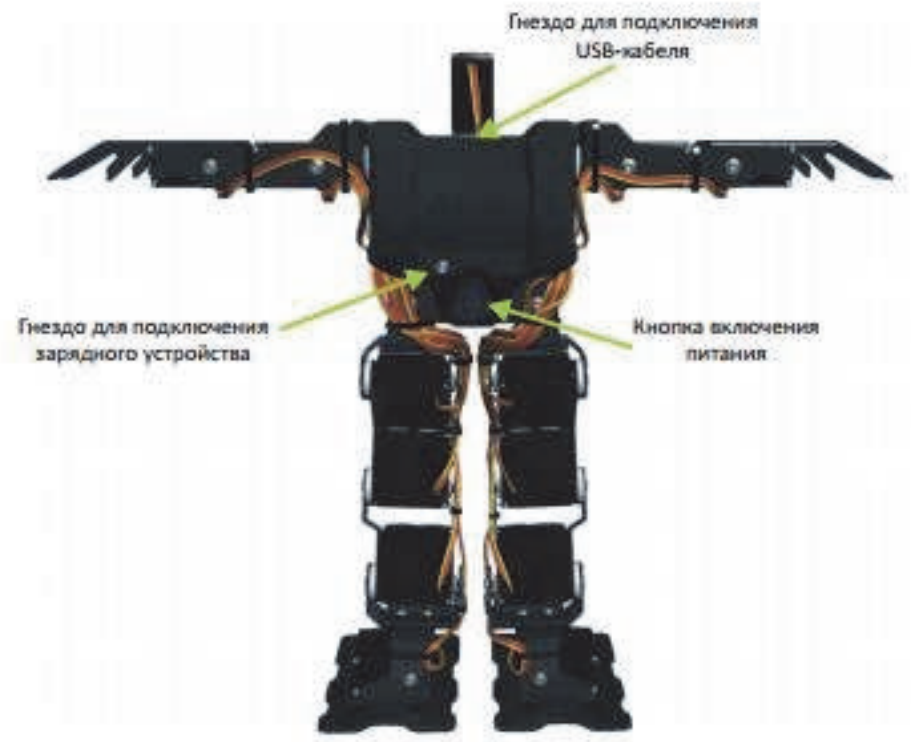


Рис. 1. Общий вид антропоморфного робота AR-101M

В модели AR-101M используется семнадцать сервомоторов (также называют сервоприводами) для управления конечностями.

Сервомоторы – это механизмы, которые отвечают за движение робота (см. рис. 2). Они отвечают за сгибание рук и ног, повороты головы и наклоны туловища.

Рис. 2. Сервопривод DS-14

Номера и группы сервоприводов

Позиция робота, представленная на рис. 3, соответствует нулевому положению сервоприводов, выставленному по умолчанию (см. рис. 4). Не рекомендуется изменять эти значения самостоятельно, поскольку это может привести к потере им равновесия при выполнении программ.

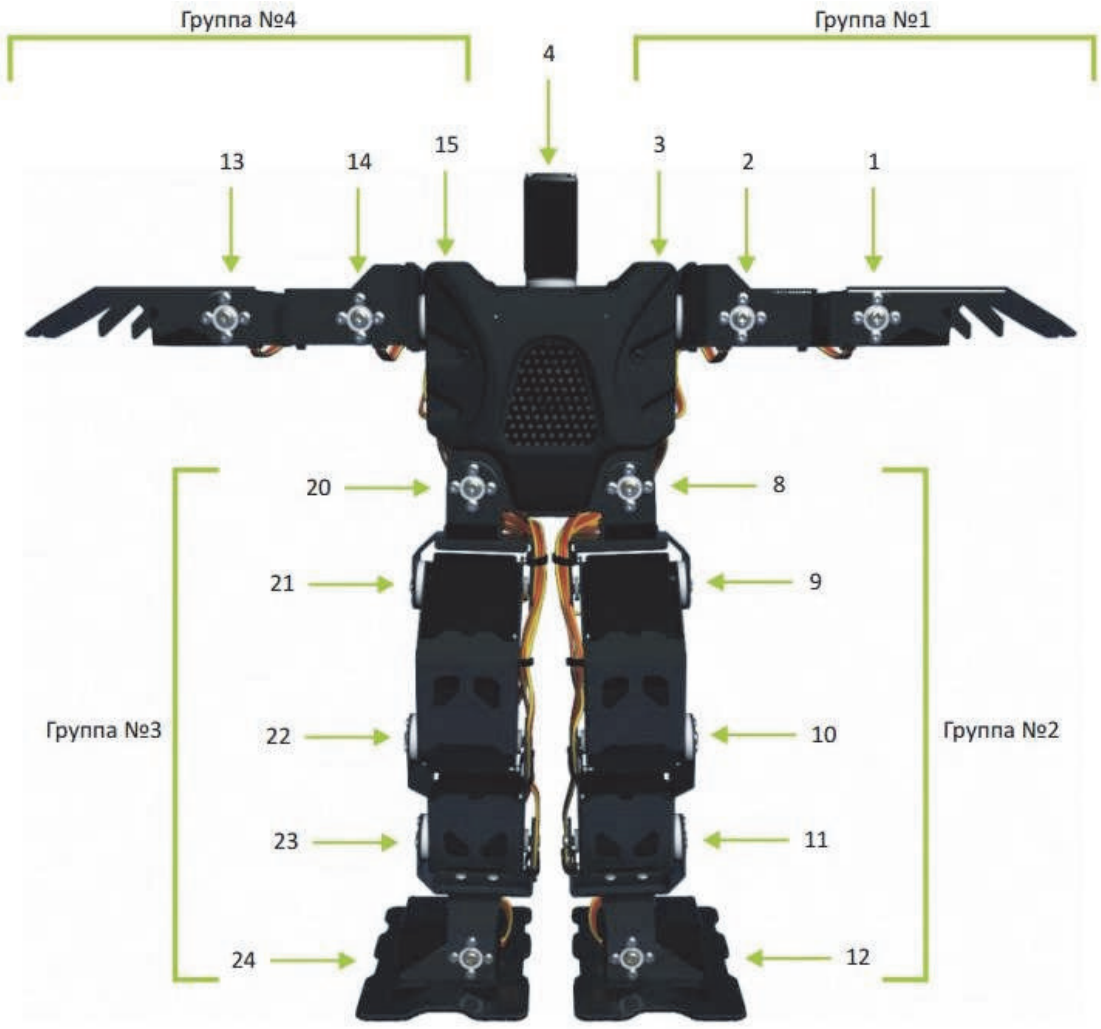


Рис. 3. Стандартное положение антропоморфного робота AR-101M

Поворот сервопривода относительно нулевого положения

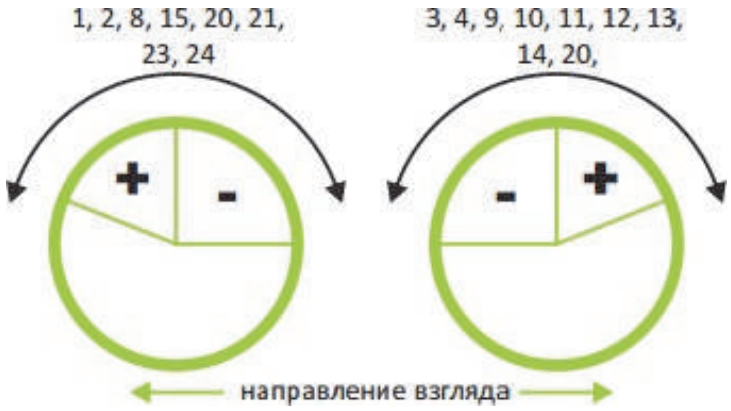


Рис. 4. Поворот сервоприводов

Рекомендуемые значения положения сервоприводов для использования в программировании, 10 =100 ед. (+/–):

- **1, 13 - +850/-850**
- **2, 14 - +900/-900**
- **3, 15 - +900/-900**
- **4 - +900/-900**
- **8, 20 - +900/-20**
- **9, 21 - +700/-850**
- **10, 22 - +870/-870**
- **11, 23 - +720/-750**
- **12, 24 - +150/-700**

Внимание! Ни в коем случае нельзя превышать данные пределы!

Микроконтроллер, или **контроллер** (МК, controller) – это микросхема для управления электронными устройствами. По сути, контроллер является мозгом робота, который получает информацию из внешнего мира с помощью сенсоров и передает управляющие сигналы в актуаторы.

Микроконтроллер с точки зрения схемотехники представляет собой однокристальный компьютер, в состав которого входит процессор, ОЗУ, ПЗУ и интерфейсы периферийных устройств. Существует множество различных микроконтроллеров, различающихся по типу процессора, объему и типу памяти, составу периферийных интерфейсов и т. п. В роботах AR-101M используется контроллер МК-66 (см. рис. 5), где количество управляемых сервоприводов в зависимости от конкретной модели может достигать 24, объем внутренней памяти – 512 Кб, количество портов ввода/вывода – 128, количество базовых операций – 128, тактовая частота процессора – 60 МГц.

Рис. 5. Контроллер МК-66

Для получения доступа к контроллеру МК-66 необходимо выполнить следующие действия:

1. Отключить провод зарядного устройства от робота.
2. Выключить питание робота.
3. Отвернуть два винта крепления плечиков (см. рис. 6).
4. Осторожно отсоединить заднюю крышку корпуса робота.

Рис. 6. Винты крепления плечиков

Зарядные устройства

В качестве источника питания используется аккумуляторная батарея. При необходимости ее замены необходимо выполнить следующие действия (см. рис. 7):

1. Выключить робота.
2. Осторожно поднять руку робота вверх, чтобы получить доступ к отсеку аккумуляторной батареи.
3. Извлечь аккумуляторную батарею из батарейного отсека.
4. Отсоединить штекер питающей батареи от контроллера.
5. Установка новой батареи производится в обратном порядке.

Рис. 7. Замена аккумуляторной батареи

Роботом можно управлять с компьютера (программное обеспечение AR-Basic Studio) либо с помощью пульта дистанционного управления. Синхронизация робота с компьютером возможна как по USB, так и через Bluetooth. Возможно расширение возможностей робота дополнительными датчиками – гироскопы, акселерометры и многие другие.

Гироскоп – это устройство со свободной осью вращения, способное реагировать на изменение угла ориентирования тела, в котором оно закрепляется. Гироскоп используется как датчик определения перемещения и поворота объекта, в котором он расположен.

Акселерометр – прибор, позволяющий определить проекцию кажущегося ускорения. В простейшем исполнении он представляет собой грузик, закрепленный на упругом подвесе. При его отклонении от первоначального положения можно определить направление изменения положения, а также величину ускорения.

Датчики играют в робототехнике одну из важнейших ролей. При помощи таких измерительных сенсоров робот лучше себя «чувствует» в пространстве. Они являются для него органами чувств, подобно человеческим глазам, ушам и другим органам восприятия мира

Выполняя все практические задания, необходимо помнить о мерах и правилах безопасности при работе с робототехническими устройствами, учитывая одновременную работу с персональным компьютером.

Внимание! При работе с окном сервомашинки будьте **ОСТОРОЖНЫ**.

Не включайте приводы, если не уверены, что робот примет безопасную позу.

Домашнее задание

1. Изучить различные виды андроидных роботов и написать эссе на тему «Возможности андроидных роботов».

Урок 2. Введение в среду программирования AR-Basic Studio

План

1. Введение в среду программирования AR-Basic Studio.
2. Изучение интерфейса.
3. Домашнее задание.

Введение в среду программирования AR-Basic Studio

Basic (сокращение от англ. Beginner's All-purpose Symbolic Instruction Code) – это семейство высокоуровневых языков программирования. Основные достоинства Basic – универсальность и легкость – сделали его популярным языком для обучения программированию.

AR-Basic Studio – это среда разработки для языка Basic от компании «Андроидные роботы» (Россия, Москва) для учебно-прикладного программирования популярных моделей андроидных роботов, таких как AR-101M. Программирование в среде AR-Basic Studio напоминает Microsoft Visual Basic и QwickBasic.

В среде программирования AR-Basic Studio есть удобный графический интерфейс. После запуска IDE (см. рис. 8) вы увидите основное окно программы, содержащее блок для написания кода, панели библиотеки и сообщений, а также главное меню.

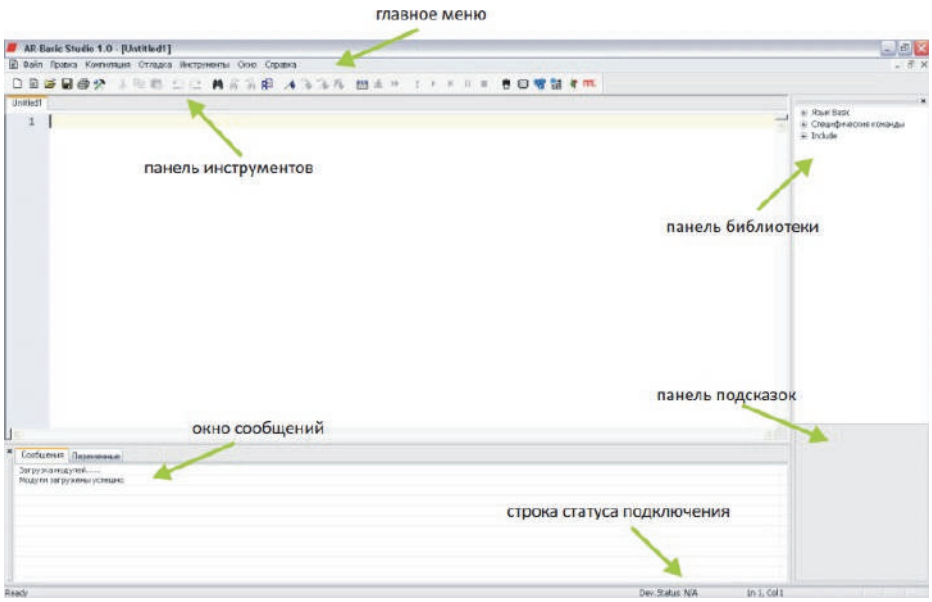


Рис. 8. Основное окно программы AR-Basic Studio

В панели библиотеки есть описание всех команд, необходимых для программирования в AR-Basic Studio в древовидной структуре. Для вставки данных из библиотеки необходимо дважды щелкнуть на нее.

В меню «Инструменты» (см. рис. 9) расположены все основные компоненты для работы с роботом – проверка соединения, управление сервомашинками и функции их отладки, а также настройки самой программы, смены COM-портов, типа контроллера. Кроме того, есть возможность настроить цветовую гамму редактора по своему усмотрению.

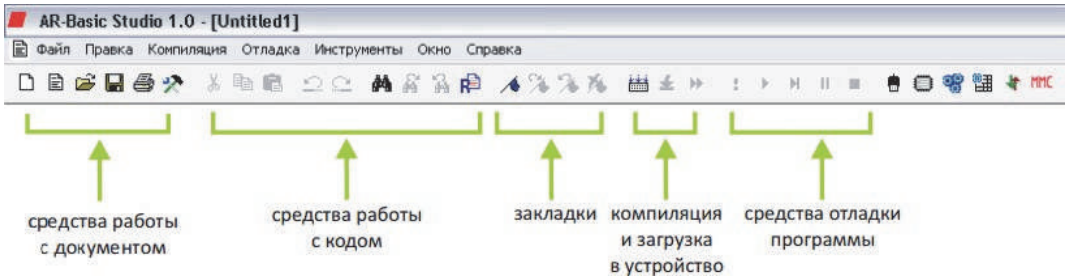


Рис. 9. Панель инструментов

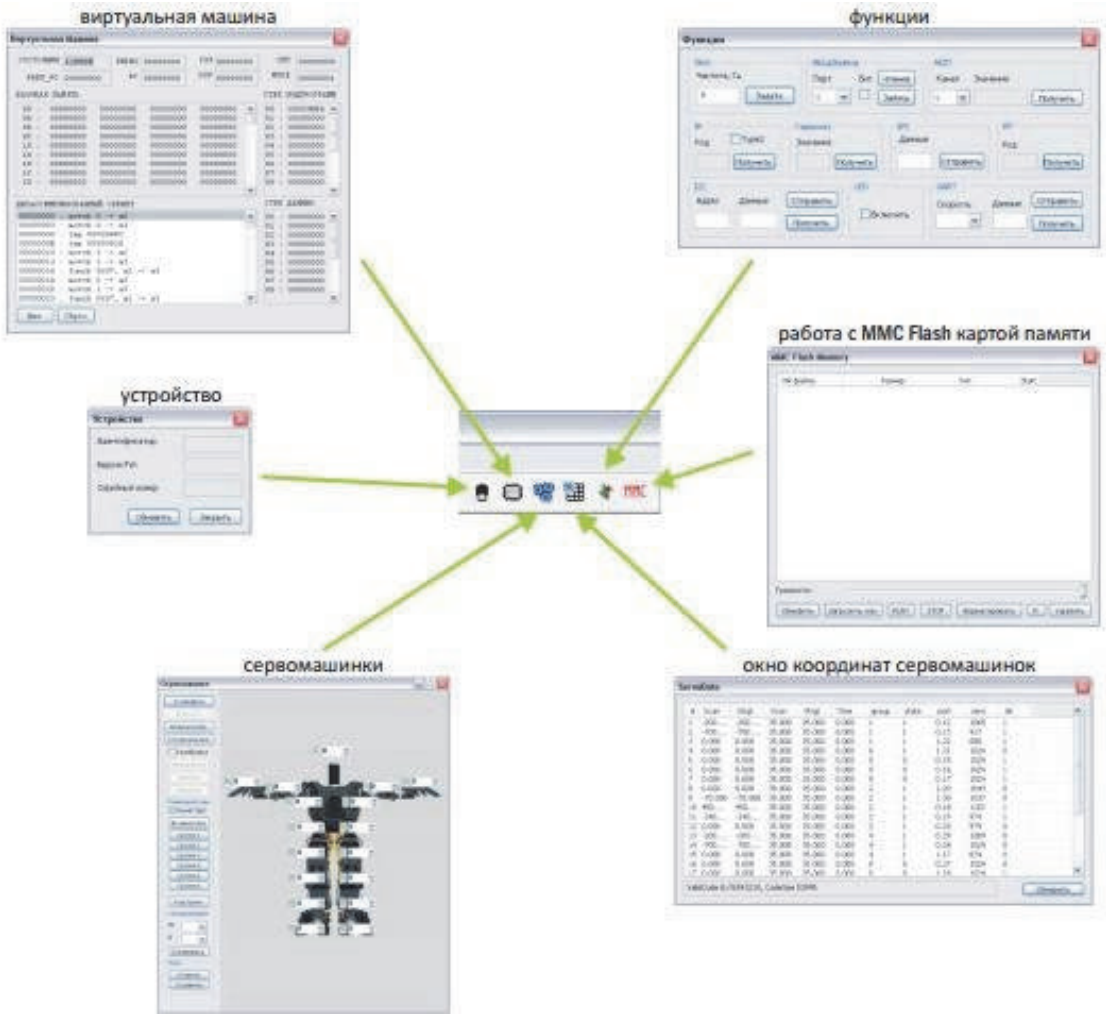


Рис. 10. Используемые окна

Информация об используемых окнах (см. рис. 10):

1) «виртуальная машина» используется для получения данных дисассемблирования программы;

2) «функции» используются в процессе отладки программы, для получения каких-либо данных, например считывания сведений с инфракрасного пульта дистанционного управления или данных с подключенных датчиков;

3) «MC-Flash Memory» служит для загрузки аудио-файлов на карту памяти робота;

4) «ServoData» (координаты сервомашинок) содержит информацию о параметрах сервомашинок;

5) «окно сервомашинки» позволяет задавать положения сервоприводов и производить вставку кода в процессе написания программы;

6) «устройство» содержит версию прошивки контроллера.

Компилирование

Компиляция – трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду.

Для того чтобы компилировать и загружать программы в устройство, необходимо сперва обеспечить связь с контроллером (раздел в панели инструментов: Инструменты – Соединение и версия устройства). Затем скомпилировать (раздел в панели инструментов: Компиляция – Компилировать) и загрузить (раздел в панели инструментов: Компиляция – Загрузить в устройство). Для запуска программы – раздел в панели инструментов: Компиляция – Запустить без отладки.

Домашнее задание

Ответьте на следующие вопросы:

1. Из каких последовательных действий состоит процесс разработки программы?

2. Для чего нужна компиляция программы?

Урок 3. Основные команды языка программирования AR-Basic Studio

План

1. Основные команды языка программирования AR-Basic Studio.

2. Домашнее задание.

Следование – выполняются компилятором в естественном порядке, начиная с первого до последнего.

Конструкция оператора следования:

```
оператор 1
оператор 2
…
```

Условный оператор – это выполнение определенных команд при условии, что некоторое логическое выражение (условие) принимает значение «истина». В большинстве языков программирования условный оператор начинается с ключевого слова if.

Конструкция условного оператора:

```
if условие then
        оператор1
else
        оператор2
end;
```

Циклический оператор While – повторяет выполнение определенных команд до тех пор, пока контрольное выражение имеет значение «условие верно». Когда контрольное выражение получает значение «условие неверно», то происходит выход из цикла.

Конструкция циклического оператора While:

```
while условие do
        оператор 1;

        оператор 2;

        ...
```

```
        оператор n;
end;
```

Циклический оператор For – позволяет повторять набор команд определенное число раз.

Конструкция циклического оператора For:

```
for параметр цикла:=1 to n do
        оператор 1;
        оператор 2;
        …
        оператор n;
end;
```

Команды среды программирования AR-Basic Studio:

goto main // переход в начало цикла.

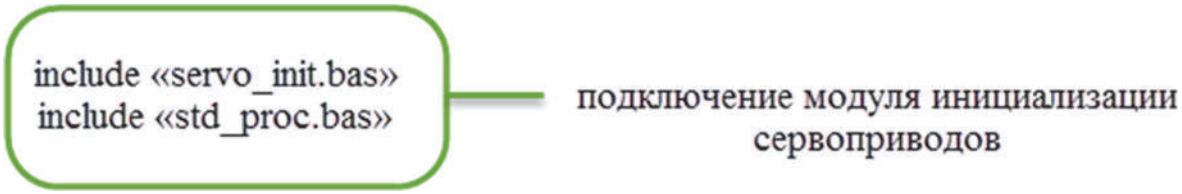
main: // начало цикла.

Строка инициализации приводов

srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0).

Подключение модулей

Дополнительные модули хранятся в файле **include**, куда можно добавлять и свои собственные. Из файлов такого типа подключаются не сами команды в файле, а только отдельные процедуры, которые далее можно будет вызывать из исходной программы. Для подключения дополнительных модулей необходимо написать ключевое слово **include** и в кавычках написать имя модуля. Например:

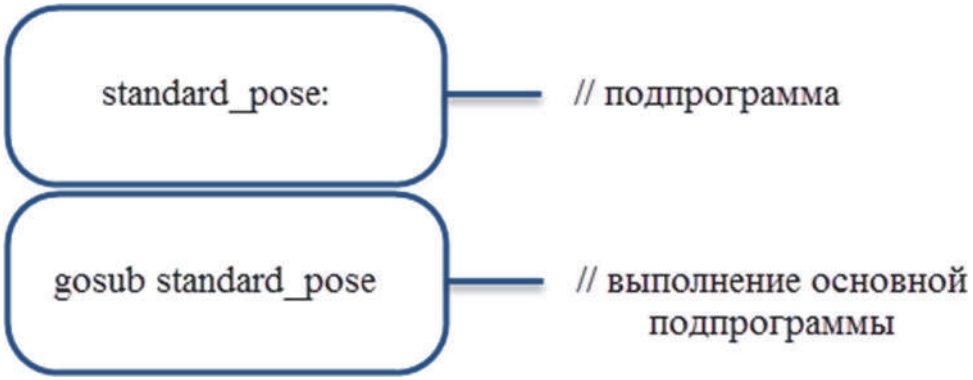


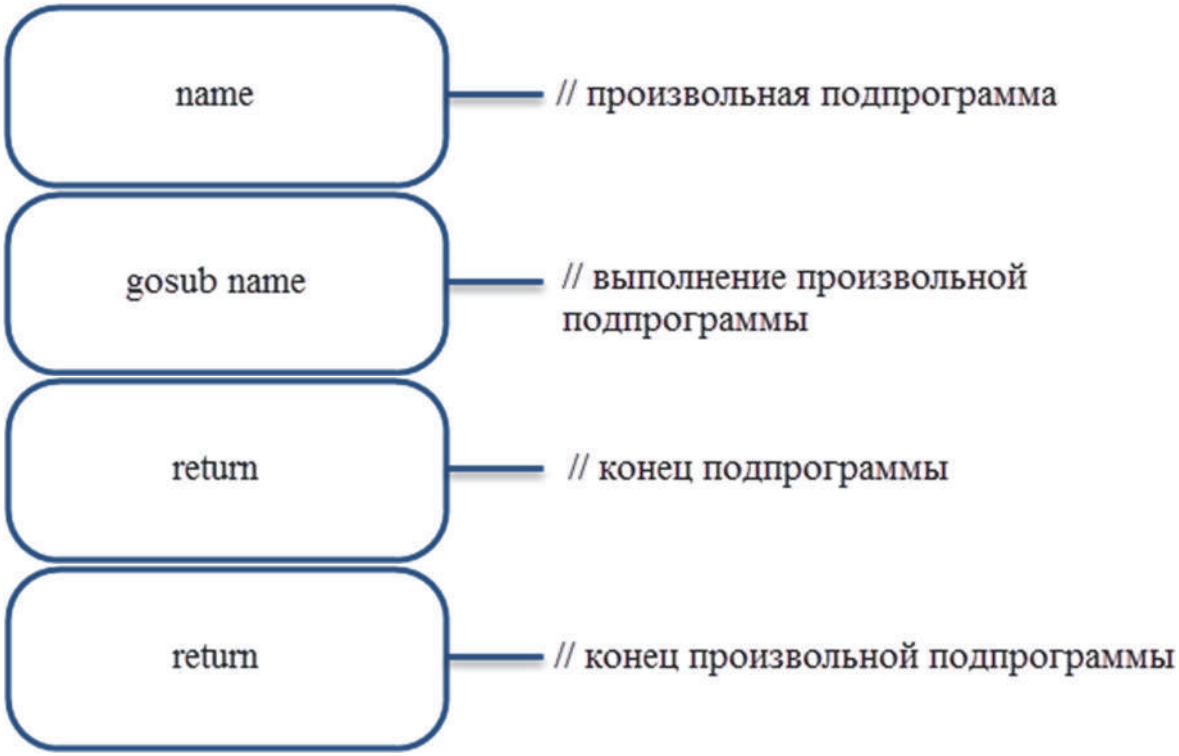
Работа с сервоприводами

srv_move_all_ptp(-200,-700,0,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,0,-70,450,-240,0) // перемещение всех сервоприводов в заданные точки.

Подпрограммы

Подпрограмма – это идентифицированная часть компьютерной программы, содержащая описание определенного набора действий. Подпрограмма может быть многократно вызвана из разных частей программы. В языках программирования для оформления и использования подпрограмм существуют специальные синтактические средства.





Обрати внимание! Далее все программы построены от простого к сложному. Изучая последующую программу, необходимо выполнить предыдущую программу.

Домашнее задание

1. Запомни основные конструкции языка программирования AR-Basic Studio (следование, условие и циклы).

Урок 4. Идентификаторы и числовые константы. Переменные. Модули

План

1. Теоретический материал. Идентификаторы и числовые константы. Переменные. Модули.
2. Домашнее задание.

Идентификаторы – любые последовательности символов из букв латинского алфавита, цифр и символов подчеркивания, начинающиеся с буквы или символа подчеркивания. Часть идентификаторов зарезервирована под ключевые слова и имена модулей. Остальные можно свободно использовать для имен переменных, констант и меток.

Числовые константы – это числа в десятичных и шестнадцатеричных системах счисления. Числа в десятичной системе записываются обычным образом, числа в шестнадцатеричной системе начинаются с префикса 0x.

Регистр символов в языке не учитывается, т. е. name, NAME, NaMe и т. п. – это одно и то же.

Ключевые слова языка

dim	const	if	then	else	end	for
to	do	while	goto	gosub	return	

Комментарии

Начинаются с последовательности // и заканчиваются концом строки.

Объявление переменных

Начинается с ключевого слова **dim**, за которым следуют через запятую (если переменных несколько) имена переменных, с инициализацией любым выражением.

dim *varname*

dim *varname = value*
dim *var1, var2, var3=100, var4, var5=var1+200*

Все переменные имеют тип int32, т. е. являются 32-битными целыми числами со знаком. Переменная, объявленная в некотором месте программы, доступна глобально в любом месте программы, в том числе и до строки с объявлением этой переменной. Инициализация переменных происходит один раз в самом начале выполнения программы (до передачи управления на первую строчку кода программы).

Объявление констант

Начинается с ключевого слова const, за которым следуют через запятую (если несколько констант) имена констант с инициализацией числами.

```
const c1 = 200
const c2 = 111, c3 = 222, c4 = 333
```

Выражения

Начинаются с имени переменной, после которой следует оператор присваивания =.

Далее следует некоторое выражение без присваивания, которое может содержать:

1) арифметические операторы: +, –, *, /;

2) логические (битовые) операторы: &, |, ^;

3) унарные операторы: –, !, ~;

4) операторы сравнения: =, >, <, >=, <=, !=, < >;

5) выражения в круглых скобках;

6) вызовы модулей с аргументами; аргументами модулей могут быть выражения без присваивания.

Арифметические операции	
+	сложение
-	вычитание
*	умножение
/	деление
Приоритет операций	
( )	выражения в скобках, вызовы модулей – наивысший приоритет
- ! ~	унарные операции
* /	умножение и деление
+ –	сложение и вычитание
< <= > >= != <>	сравнение
& | ^	логические битовые операции

Операции сравнения

Операторы сравнения =, <, >, <= и >= сравнивают выражение в левой части оператора с выражением в правой части оператора и представляют результат в виде логического значения True или False. Например:

5 > 3 = True; «A» = «B» = False

Операции сравнения	
<	меньше
<=	меньше или равно
>	больше
>=	больше или равно
!= ; <>	неравно
=	равно

Логические операции

Над элементами логических выражений могут производиться логические операции, которые на языках программирования обозначаются следующим образом: логическое умножение (конъюнкция), логическое сложение (дизъюнкция), логическое отрицание, логическое исключающее ИЛИ.

Логические операции	
&	логическое И (конъюнкция)
|	логическое ИЛИ (дизъюнкция)
^	логическое ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR)

Унарные операции

Унарные операции применяются к операндам, стоящим справа от них.

Унарные операции	
-	смена знака операнда
!	логическая инверсия; если операнд был ненулевым, результатом операции будет ноль, иначе результатом будет единица
~	битовая инверсия; инвертирует число побитно

Модуль – это часть программы, которая может быть скомпилирована отдельно. В дальнейшем готовые скомпилированные модули (объектные модули) могут объединяться в готовые программы или библиотеки. В модулях обычно хранятся подпрограммы, константы или переменные.

Вызовы модулей

Модуль имеет имя, список аргументов и возвращаемое значение. Теоретически модуль может читать данные из возвращаемого значения и изменять аргументы, но это имеет смысл только в случае использования переменных в качестве аргументов и для записи возвращаемого значения, в случае выражений.

Синтаксис вызова модуля:

```
module1()

module2(var1, var2)

var1= some_module2(var2, var3+var4, 100, var5-200*var6)
```

Домашнее задание

Ответь на следующие вопросы:

1. Что такое идентификатор и сколько произвольных символов может содержать?

2. В Бейсике существует пять категорий операций, перечисли их.

3. Что такое модуль и как осуществляется вызов модуля?

Урок 5. Условия. Циклы. Подпрограммы. Процедуры и функции

План

1. Теоретический материал. Циклы. Условия. Подпрограммы. Процедуры и функции.
2. Домашнее задание.

Оператор ветвления IF-THEN-ELSE

Общая форма следующая:

```
if условие1 then
        оператор1
else if условие 2 then
        оператор 2
        ...
else if условие N then
        оператор N
else
        оператор
end;
```

Здесь условие – это любое выражение без присваивания. Блок **else** в конце, а также блоки **else if** могут отсутствовать. В конце блока **if** в любом случае обязательно ставится **end**. Условие проверяется перед выполнением оператора.

Управление передается на блок **if**, если условие блока ненулевое выполняется (не равно нулю). Если условие не выполняется (равно нулю), то проверяется условие следующего блока **else if** (если таковые имеются), или управление передается на блок **else** (если таковой имеется), или управление передается на оператор, следующий за оператором **end**. Если какой-то блок **if** выполнился, управление передается на оператор после оператора **end**, другие условия не проверяются.

Оператор цикла WHILE – DO

Общая форма:

```
while условие do
    оператор
end
```

Здесь условие – это любое выражение без присваивания.

Оператор цикла FOR – DO

Общая форма:

```
for параметр цикла=начальное значение to конечное значение do
    оператор
end
```

Здесь параметр цикла – объявленная переменная-счетчик, начальное значение – выражение без присваивания, используемое для начальной инициализации счетчика, конечное значение – выражение без присваивания, при достижении которого код цикла будет выполнен последний раз.

Счетчик инициализируется каждый раз при входе в цикл из внешнего кода; проверка на выход из цикла осуществляется после каждой итерации цикла; проверяется условие «параметр цикла ≥ конечное значение», при выполнении этого условия работа цикла прекращается и управление передается на оператор, следующим за циклом. В конце каждой итерации цикла счетчик увеличивается на единицу.

Оператор безусловного перехода GOTO и метки

Метка – это уникальный идентификатор (не совпадающий с именами переменных и констант), после которого следует двоеточие. Метка объявляется в любом месте кода и доступна глобально во всей программе, в том числе и до строки, в которой она объявлена. Метки используются для передачи управления на код, который следует непосредственно за объявлением метки.

Label: code

Здесь code – любой код (выражение с присваиванием, вызов модуля или любой оператор).

Оператор **goto** передает управление на метку, которая указывается после ключевого слова **goto**: goto Label.

Операторы перехода на подпрограмму **gosub** и возврата из подпрограммы **return**. Оператор **gosub** предназначен для передачи управления на некоторую метку с сохранением адреса перехода. Если в коде, на который было передано управление с помощью **gosub**, встретится оператор **return**, управление будет передано на оператор, следующий непосредственно за оператором **gosub**. Таким образом, можно создавать подпрограммы. Максимальная глубина вложенности вызовов подпрограмм равна 16.

Подпрограммы

В языках программирования вспомогательные алгоритмы называются подпрограммами. В AR-Basic Studio различаются две разновидности подпрограмм – процедуры и функции. Процедуры и функции представляют собой отдельные блоки, из которых складывается код программы, каждая подпрограмма выполняет какую-то задачу или ее часть.

Функция – это подпрограмма, которую вызывают, чтобы выполнить какие-то расчеты или проверки. Когда она завершает работу, то возвращает управление вызывающей программе и передает ей результат расчета.

Процедура – это тоже подпрограмма. Ее тоже вызывают, чтобы выполнить какие-то действия, но от нее не требуется возвращать основной программе какие-либо значения.

Процедура начинается с оператора **gosub** и заканчивается **gosub standard_pose**, между которыми и помещается код. Процедуры могут вызываться или самим Visual Basic, например реализованные в виде обработчиков событий, или другими процедурами и функциями. Имя процедуры обработки события состоит из имени объекта, знака подчеркивания и имени события:

```
gosub

gosub standard_pose

return
```

Домашнее задание

Ответь на следующие вопросы:

1. Что такое цикл и для чего он нужен?
2. Чем отличаются процедуры от функций?
3. Как осуществляется вызов процедуры?

Урок 6. Изучение и выполнение программы: работа с одним сервоприводом

План

1. Теоретический материал. Изучение и выполнение программы: работа с одним сервоприводом.
2. Экспериментальная часть. Напишите готовую программу и запустите ее.
3. Самостоятельная работа. Выполните упражнения 1 и 2. Результаты покажите учителю.

Приступая к написанию программ, следует:

1) установить робота на ровной горизонтальной, свободной от посторонних предметов поверхности (площадью не менее 1 м^2);

2) подключить робота к компьютеру с помощью USB-кабеля;

3) включить питание робота;

4) открыть программу AR-Basic Studio;

5) проверить статус соединения робота с компьютером, в нижней строке состояния соединения должна быть надпись Online (см. рис. 11).

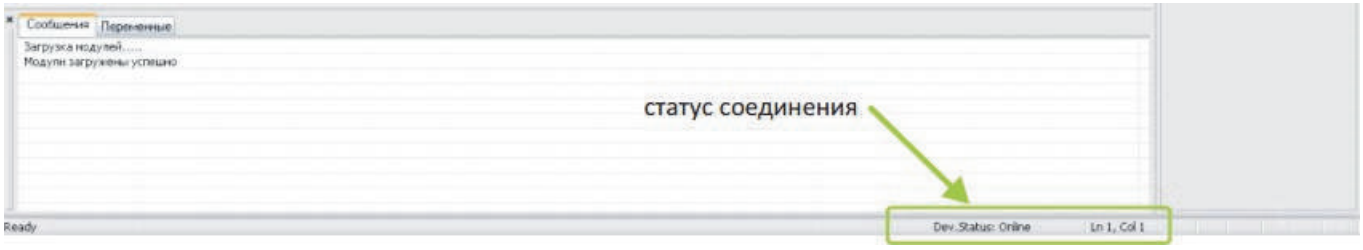


Рис. 11. Статус соединения

Для настройки соединения зайдите в Свойства системы (клавиша Windows+Pause, правый щелчок на Мой компьютер – Свойства или через Панель управления – Система). Выберите вкладку Оборудование и нажмите Диспетчер устройств.

Зайдите в раздел COM-Порты и найдите устройство USB Serial Port, номер используемого COM-порта должен соответствовать выбранному COM-порту AR-Basic Studio-Настройка-Общие-Порт.

После установки связи с роботом через компьютер можно приступать к написанию программы для управления роботом, используя для установки значений для сервоприводов окно «сервомашинки».

Языковые конструкции AR-Basic позволяют управлять сервоприводами как по отдельности, так и объединяя их в группы. Принцип управления прост: большинству команд в качестве параметров передаются лишь номер сервомотора или группы сервомоторов и значения, на которые должен подвинуться тот или иной сервопривод. Есть также команды для обмена данными – приема сигналов от пульта ДУ или компьютера, передачи данных по каналу Bluetooth, подачи звуковых и световых сигналов.

Ограничения сервопривода робота AR-100 составляют от 0 до 2048 единиц. Для удобства при выставлении нулевой отметки за максимальное значение принято брать число 1024, а за минимальное – минус 1024. Таким образом, программисту доступен диапазон значений примерно от –1000 до 1000, где 0 – середина.

В качестве параметров к этим функциям задается числовое значение – идентификатор привода (или группы) и одна или несколько точек, определяющих, куда следует повернуть конкретный привод (от –1000 до 1000).

Простейшая команда srv_move(mn, val) означает движение привода с номером mn в позицию val с учетом текущей скорости. А скорость можно задать командами srv_speed(mn, val) (для одного мотора с номером mn) или srv_speed_all(val) – для всех сразу. Интересна команда srv_moveptp(mn, val), обеспечивающая перемещение мотора с номером mn в позицию val таким образом, чтобы все сервомоторы в той же группе, что и mn, закончили движение одновременно. Вообще ptp (point-to-point) в названии функции означает автоматическое смещение скоростного режима для некоторых приводов в группе или вцелом, для того чтобы конечное положение приводы в группе (или в целом) приняли одновременно.

Предположим, приводы плеча и локтевого сустава находятся в нейтральном положении, и мы хотим заставить робота поднять согнутую в локте руку. Для этого нам нужно подвинуть соответствующие приводы в разные точки, например один в точку 300, а другой – в 600. Но поскольку скорость движения приводов по умолчанию одинакова, второй будет двигаться в два раза дольше. Чтобы избежать этого, нужно использовать команду PTP; тогда контроллер автоматически выставит для первого привода скорость в два раза меньше и оба сустава достигнут своих конечных точек одновременно.

Конечно, сразу научить робота определенным действиям довольно непросто. Чего стоит только разобраться в движении сервоприводов! Но разработчики AR-100 предусмотрели более простой способ программирования – визуальный. В AR-Basic Studio есть специальное окно управления сервоприводами (см. рис. 12), где на изображении робота рядом с каждым сервомотором указаны параметры: индикатор включения и число, характеризующее движение сервопривода. Это же окно позволяет автоматически генерировать фрагменты кода, описывающие определенные позы робота (поклон, приседание, поднятие рук и т. д.).

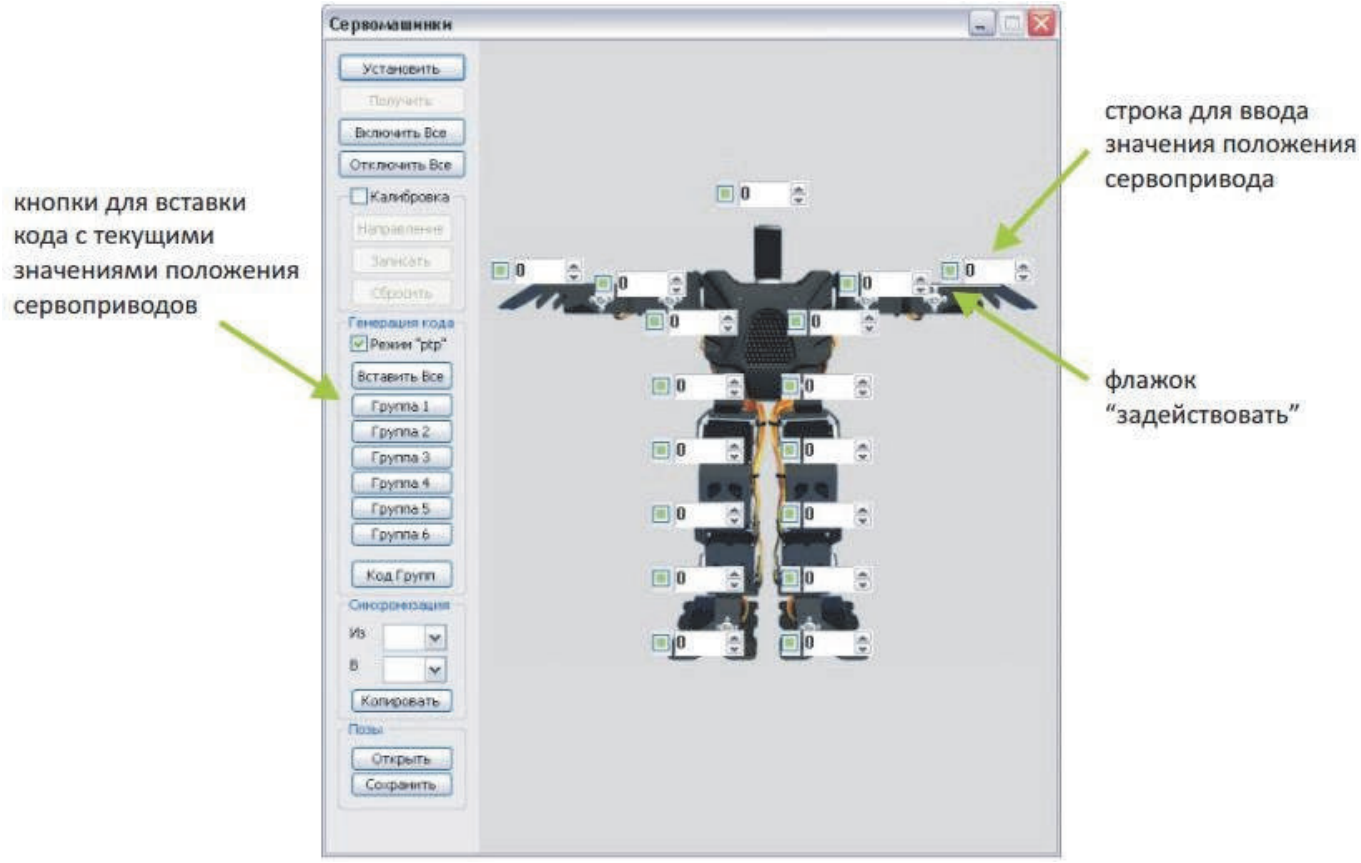


Рис. 12. Окно сервомашинки

Если робот подключен к компьютеру, то можно сразу же проверить правильность заданных команд. Синхронизация робота с компьютером возможна как по USB, так и через порт Bluetooth.

Экспериментируй

Программа для работы с одним сервоприводом. Выполнение поворота головы

В ряде случаев при выполнении сложных движений возникает необходимость индивидуального управления каким-либо приводом с помощью команды srv_init (mn, val). В данном уроке практическое применение команд

индивидуального управления отдельным сервоприводом будет показано на примере поворота головы робота в одну сторону.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
   include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

main: //Главная программа
    gosub standard_pose
    gosub golova
stop
    goto main //переход в начало цикла

standard_pose: //Стандартная поза
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,
-240,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидание завершения движение всех приводов
return  //Конец процедуры
golova: //Название (начало) процедуры
    srv_speed_all(6000) //Установить скорости для всех приводов
    srv_move(4,600) //Перемещение заданного привода в заданное положение (№ привода, позиция)
    srv_waitmove_all()
    gosub standard_pose //Стандартная поза
return //Конец процедуры
```

Перед записью программы в контроллер робота, ее необходимо сохранить (Файл → Сохранить или щелкнуть левой кнопкой мыши по соответствующей иконе в строке инструментов). После этого на панели инструментов нажмите кнопку «компилировать, загрузить и запустить программу». Если в коде не использован оператор stop, то после загрузки программы в контроллер робот приступит к ее выполнению. Для запуска вышенаписанной программы, на панели инструментов нажмите кнопку «записать с отладкой».

Средства отладки программы, позволят: запустить программу без отладки, запускать программу с отладкой, пошагово выполнить действия, приостановить выполнение программы и полностью остановить выполнение программы.

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он поворачивал голову в противоположную сторону.

2. Запрограммируйте робота так, чтобы он поворачивал голову в обе стороны.

Урок 7. Изучение и выполнение программ: работа с группой сервоприводов. Поднятие руки

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнение 1 и 2. Результаты покажите учителю.

Экспериментируй

В данном уроке будет рассмотрено использование команды srv_moveptpg3. Используйте для вставки кода в программу окно «сервомашинки». В данном примере робот поднимает руку.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

//Главная программа
 main:
    gosub standard_pose
     gosub ruka
stop
    goto main //переход в начало цикла
```

```
//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидание завершения движение всех приводов
return  //Конец процедуры


ruka: //Название (начало) процедуры
    srv_speed_all(6000) //Установить скорости для всех приводов
    srv_move(14,900) //Перемещение заданного привода в заданное положение (№ привода, позиция)
    srv_waitmove_all()
    gosub standard_pose
return //Конец процедуры
```

Несмотря на то что код программы выполняется роботом последовательно, при написании сложных движений следует учитывать текущие занимаемые роботом положения и скорость перемещения сервоприводов. Не задавайте в группе команд srv_move, srv_movetpg, srv_move_all случайно взятые значения, поскольку это может привести к потере равновесия робота или остановке сервоприводов при взаимном их пересечении.

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он поднимал руку параллельно полу.

2. Напишите программу, чтобы робот поднимал и опускал руку без использования команды gosub standard_pose.

Урок 8. Изучение и выполнение программ с применением нескольких операций. Подъем руки со сгибанием в локте

План

1. Экспериментальная часть. Напишите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнение 1 и 2. Результаты покажите учителю.

Экспериментируй

В данном уроке мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, и применение нескольких операций, которое будет показано на примере: подъем руки со сгибанием в локте.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)


//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»


//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу


//Главная программа
main:
    gosub left_hand_up
    gosub standard_pose
stop
```

```
    goto main //переход в начало цикла


//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,
-240,0)//Перемещение всех приводов в заданные позиции с одновременным завершением
движения
    srv_waitmove_all()//Ожидание завершения движение всех приводов
 return //Конец процедуры


left_hand_up:
    srv_speed_all(1000) //Установить скорости для всех приводов
    srv_moveg3(1, -900, -500, 650) // синхронное перемещение сервоприводов группы № 1
tim_delay(6000) // задает задержку дальнейшего выполнения программы в мл.с.
     srv_moveptpg3(1, -200, -700, 0)
    srv_waitmove_all() //Ожидание завершения движение всех приводов
 return //Конец процедуры
    srv_waitmove_all() //Ожидание завершения движение всех приводов
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он прижал руку к голове.
2. Запрограммируйте робота так, чтобы он прижал руку к груди.

Урок 9. Изучение и выполнение программ с применением нескольких операций. Приседание

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнение 1. Результаты покажитеучителю.

Экспериментируй

В данном уроке мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, и применение нескольких операций, которое будет показано на примере выполнении приседания роботом.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)


//Подключаем необходимые модули
     include «servo_init.bas»
    include «std_proc.bas»


//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу


//Главная программа
 main:
    gosub standard_pose
    gosub prisedanie
stop
```

```
    goto main //переход в начало цикла
//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


prisedanie: //Название (начало) процедуры
    srv_speed_all(2000) //Задаем скорость всем сервоприводам
    srv_moveptpg5(3, 10,400,-770,510,0) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg5(2, 10,400,-770,510,0)
    srv_moveptpg3(4, -200,-700,0)
    srv_moveptpg3(1, -200,-700,0)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return  //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он приседал несколько раз (выполнить программу с помощью цикла).

Урок 10. Изучение и выполнение программ: применение сложных операций. Выполнение поклона

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.
2. Самостоятельная работа. Выполните упражнение 1 и покажите учителю.

Экспериментируй

В данном уроке мы освоим практическое применение команд, в том числе применение сложных операций, которое будет показано на примере выполнения поклона роботом.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)


//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»


//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу


//Главная программа
main:
    gosub standard_pose
    gosub poklon
stop
    goto main //переход в начало цикла
```

```
//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
 return //Конец процедуры


poklon: //Название (начало) процедуры
    srv_speed_all(3000) //Задаем скорость всем сервоприводам
    srv_moveptpg5(2, 0, 200, 400, -280, 0) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg5(3, 0, 200, 400, -280, 0)
    srv_waitmove_all()
tim_delay(100)
    srv_moveg3(1, -900, -500, 650)
    srv_move(4, 150)
    srv_waitmove_all()
tim_delay(400) //Задержка приводов на определенное время
    srv_moveptpg3(1, -200, -700, 0)
    srv_moveptpg3(4, -200,-700,0)
    srv_moveptpg3(1, -200,-700,0)
    srv_waitmove_all()
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он вытянул руки в сторону и выполнил поклон.

Урок 11. Изучение и выполнение программ: применение сложных операций. Установка робота на одну ногу

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.
2. Самостоятельная работа. Выполните упражнение 1 и покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере установки робота на одну ногу.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

//Главная программа main:
    gosub standard_pose
    gosub ustanovka_na_nogu
 stop
    goto main //переход в начало цикла

//Стандартная поза
```

```
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


ustanovka_na_nogu: //Название (начало) процедуры
    srv_speed_all(1500) //Задаем скорость всем сервоприводам
    srv_move_all_ptp(-200,-600,0,0,0,0,0,120,-90,520,-290,150,-200,-600,0,0,0,0,0,-80,-70,450,-240,-120) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
     srv_speed_all(2300)
    srv_move_all_ptp(500,500,0,0,0,0,0,150,150,50,-20,100,500,500,0,0,0,0,0,-70,- 20,450,-260,-160)
    srv_waitmove_all()
tim_delay(1000) //Задержка приводов на определенное время
     srv_speed_all(2300)
    srv_speed_all(1500)
    srv_move_all_ptp(500,500,0,0,0,0,0,150,150,50,-20,100,500,500,0,0,0,0,0,-70,- 20,450,-260,-160)
    srv_waitmove_all()
    srv_move_all_ptp(-200,-600,0,0,0,0,0,120,-90,520,-290,150,-200,-600,0,0,0,0,0,-80,-70,450,-240,-120)
    srv_waitmove_all()
return //Конец процедуры
```

Выполни самостоятельно

1. Установить робота на одну ногу, затем на другую.

Урок 12. Изучение и выполнение программ: применение сложных операций. Ходьба вперед

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнение 1 и 2. Результаты покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнения ходьбы вперед роботом.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

//Главная программа
main:
    gosub standard_pose
    gosub forward_walk
stop
    goto main //переход в начало цикла
```

```
//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


forward_walk: //Название (начало) процедуры
    srv_speed_all(2500) //Задаем скорость всем сервоприводам
    srv_move_all_ptp(-200,-600,0,300,0,0,0,120,-90,520,-290,150,-200,-600,0,0,0,0,0,-80,-70,450,-240,-120) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_speed_all(9000)
    srv_move_all_ptp(-200,-600,-100,300,0,0,0,140,50,50,70,100,-200,-600,0,0,0,0,0,-100,-70,450, -240,-140)
    srv_waitmove_all()
    srv_move_all_ptp(-200,-600,-200,300,0,0,0,140,220,430,-440,100,-200,-600,50,0,0,0,0,-100,-100,450,-200,-130)
    srv_waitmove_all()
    srv_move_all_ptp(-200,-600,-200,0,0,0,0,140,120,630,-540,100,-200,-600,50,0,0,0,0,-100,-100,450,-200,-120)
    srv_waitmove_all()
    srv_speed_all(3500)
    srv_move_all_ptp(-200,-600,-100,-300,0,0,0,0,130,410,-340,0,-200,-600,0,0,0,0,0,0,-200,560,-170,0)
    srv_waitmove_all()
    srv_move_all_ptp(-200,-600,0,-300,0,0,0,-100,50,420,-220,-130,-200,-600,0,0,0,0,0,140,-150,360,20,100)
    srv_waitmove_all()
    srv_speed_all(9000)
    srv_move_all_ptp(-200,-600,0,-300,0,0,0,-100,-70,450,-240,-130,-200,-600,-100,0,0,0,0,140,50,50,70,100)
```

```
 srv_waitmove_all()
     srv_move_all_ptp(-200,-600,50,-300,0,0,0,-100,-100,450,-200,-130,-200,- 600,-
200,0,0,0,0,140,220,430,-440,100)
     srv_waitmove_all()
     srv_move_all_ptp(-200,-600,50,0,0,0,0,-100,-100,450,-200,-120,-200,-600,-
200,0,0,0,0,140,120,630,-540,100)
     srv_waitmove_all()
     srv_speed_all(3500)
     srv_move_all_ptp(-200,-600,0,300,0,0,0,0,-200,560,-170,0,-200,-600,-100,0,0,0,0,0,130,410,-
340,0)
     srv_waitmove_all()
     srv_move_all_ptp(-200,-600,0,300,0,0,0,120,-150,360,20,200,-200,- 600,0,0,0,0,0,-
110,50,420,-220,-120)
     srv_waitmove_all()
     srv_speed_all(9000)
     srv_move_all_ptp(-200,-600,-100,300,0,0,0,120,50,50,70,100,-200,- 600,0,0,0,0,0,-110,-
70,450,-240,-120)
     srv_waitmove_all()
     srv_speed_all(2500)
     srv_move_all_ptp(-200,-600,0,0,0,0,0,120,-90,520,-290,150,-200,-
     600,0,0,0,0,0,-80,-70,450,-240,-110)
     srv_waitmove_all()//Ожидания завершения предшествующих действий
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он выполнил поклон со сгибанием в коленях.

2. Запрограммируйте робота так, чтобы он выполнил ходьбу назад.

Урок 13. Изучение и выполнение программ с использованием циклов и условий для движения

План

1. Теоретический материал. Изучение и выполнение программ с использованием циклов и условий для движения.

2. Экспериментальная часть. Запрограммируйте робота так, чтобы он поворачивал голову несколько раз.

Прочитай

Циклы используются для организации повторного выполнения блоков кода. Любой цикл состоит из двух частей – условие цикла и тело цикла. У любого цикла есть параметр. Параметр цикла – это переменная, которая изменяется в теле цикла, а также участвует в условии его окончания.

Цикл for с определенным количеством повторений. Цикл выполняется от начального до конечного значения параметра с заданным шагом.

Совокупность простых циклов, вложенных один в другой, называется сложным (вложенным) циклом. При конструировании сложных циклов необходимо руководствоваться следующими правилами:

1) нельзя войти во внутренний цикл, минуя вход внешнего цикла;

2) имена параметров простых циклов не должны повторяться в конструкции сложного цикла;

3) простые циклы не должны пересекаться в конструкции сложного цикла, т. е. окончание внешнего цикла не должно предшествовать окончанию внутреннего цикла.

Экспериментируй

*Пример кода подпрограммы: с использованием цикла **for**:*

```
povorot_golovi:
    for i=1 to 3 do
        srv_move(4,500)
        srv_waitmove_all()
        srv_move(4,-500)
        srv_waitmove_all()
        rv_move(4,0)
        srv_waitmove_all()
```

```
    end
return
```

*С использованием цикла **while**:*

В начале программы необходимо объявить переменную i:

```
dim i=0
povorot_golovi_2:
    while i<3 do
        srv_move(4,500
        srv_waitmove_all()
        srv_move(4,-500)
        srv_waitmove_all()
        srv_move(4,0)
        srv_waitmove_all()
        i=i+1
    end
return
```

С использованием условного оператора **if**

else:

```
if v = 0x0002 then
    gosub standard_pose
    srv_speed_all(3500)
else if v = 0x0007 then
    gosub bow_pose
    gosub standard_pose
end
```

Запрограммируйте робота так, чтобы он поворачивал голову несколько раз.

Домашнее задание

Ответь на следующие вопросы:

1. Что такое цикл и для чего он нужен?

2. В чем основное отличие между циклами с предусловием и с постусловием?

3. Что такое сложный цикл и каковы основные правила его конструирования?

Урок 14. Изучение и выполнение программ: работа с базой подпрограмм. Барабанщик

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнения 1 и 2. Результаты покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами с использованием подпрограмм, которое будет показано на примере выполнении команды «барабанщик».

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

//Главная программа
main:
    gosub standard_pose
    gosub cut
stop
    goto main //переход в начало цикла
```

```
//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидание завершения движение всех приводов
return //Конец процедуры


cut: //Название (начало) процедуры
    srv_speed_all(12000) //Установить скорости для всех приводов
    srv_moveptpg3(1,-200,-700,900) // используются только групповые движения трех моторов на руках
    srv_moveptpg3(4, -200, -700, 900)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
tim_delay(50) //задержка приводов на определенное время
    srv_move(4,-200) //Перемещение заданного привода в заданное положение (№ привода, позиция)
    srv_moveptpg3(1, -200, -900, 900)
    srv_moveptpg3(4, -200, -900, 500)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_move(4, 200) //Перемещение заданного привода в заданное положение (№ привода, позиция)
    srv_moveptpg3(4, -200, -900, 900) // используются только групповые движения трех моторов на руках
    srv_moveptpg3(1, -200, -900, 500)
    srv_waitmove_all()
    srv_move(4, -200)
    srv_moveptpg3(1, -200, -900, 900)
    srv_moveptpg3(4, -200, -900, 500)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
    srv_move(4, 200) //Перемещение заданного привода в заданное положение (№ привода, позиция)
    srv_moveptpg3(4, -200, -900, 900) // используются только групповые движения трех моторов на руках
    srv_moveptpg3(1, -200, -900, 500)
```

```
    srv_waitmove_all()
    srv_move(4, -200)
    srv_moveptpg3(1, -200, -900, 900)
    srv_moveptpg3(4, -200, -900, 500)
    srv_waitmove_all()
    srv_move(4, 200)
    srv_moveptpg3(4, -200, -900, 900)
    srv_moveptpg3(1, -200, -900, 500)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_moveptpg3(1, -200, -700, 900)
    srv_moveptpg3(4, -200, -700, 900)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    gosub standard_pose //Ставим робота в стандартную позу
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он похлопал руками.
2. Запрограммируйте робота так, чтобы он махал руками, как птица.

Урок 15. Изучение и выполнение программ: работа с базой подпрограмм. Прыжок

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнения 1 и 2. Результаты покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере прыжка робота на правой и на левой ноге.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

//Главная программа
main:
    gosub standard_pose
    gosub jump
stop
    goto main  //переход в начало цикла
```

```
//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


jump: //Название (начало) процедуры
    srv_moveptpg5(3, 0,300,-760,620,150) //используются только групповые движения пяти моторов на ногах
    srv_moveptpg5(2, 0,300,-760,620,150)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
    srv_moveptpg5(3, 0,300,-760,620,0)
    srv_moveptpg5(2, 0,300,-760,620,0)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он поднял ступни по очереди.

2. Запрограммируйте робота так, чтобы он прыгнул сначала на левой ноге, затем на правой.

Урок 16. Изучение и выполнение программ: работа с базой подпрограмм. Казачок

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнения 1 и 2. Результаты покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнении команды казачок.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)


//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»


//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу


//Главная программа
main:
    gosub standard_pose
    gosub d23_kazachok_start
    gosub d23_kazachok
stop
    goto main  //переход в начало цикла
```

```
//Стандартная поза
standard_pose:
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all()//Ожидания завершения предшествующих действий
return //Конец процедуры
d23_backward_standup: //Название (начало) процедуры
    srv_speed_all(5500) //Задаем скорость всем сервоприводам
    //srv_move_all_ptp(-900,300,0,0,0,0,0,0,150,0,-900,0,- 900,300,0,0,0,0,0,0,150,0,-900,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    //srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_move_all_ptp(-950,300,-900,0,0,0,0,0,400,-370,-950,0,-900,300,-900,0,0,0,0,0,400,-370,-900,0)
    srv_waitmove_all()
    srv_move_all_ptp(-200,-900,-900,0,0,0,0,0,-500,-550,260,0,-200,-900,- 900,0,0,0,0,0,-500,-550,260,0)
    srv_waitmove_all()
    srv_move_all_ptp(50,-800,-900,0,0,0,0,0,-850,-300,650,0,50,-800,-900,0,0,0,0,0,- 850,-300,650,0)
    srv_waitmove_all()
    srv_move_all_ptp(-500,-300,100,0,0,0,0,0,0,-600,650,0,-500,- 300,100,0,0,0,0,0,0,-600,650,0)
    srv_waitmove_all()
    srv_speed_all(5000) //Задаем скорость всем сервоприводам
return //Конец процедуры


d23_kazachok: //Название (начало) процедуры
for i = 1 to 16 do
    srv_moveptpg5(3, 10, 500, -770,-275, 0) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg5(2, 10, -300, -770, 510, 0)
    srv_waitmove_all()
    srv_moveptpg5(2, 10, 500, -770, -275, 0)
```

```
    srv_moveptpg5(3, 10, -300, -770, 510, 0)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
end
    gosub d23_backward_standup
return //Конец процедуры
d23_kazachok_start: //Название (начало) процедуры
    gosub d23_sitdown
    srv_moveptpg3(1, -200, -700, -900) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg3(4, -200, -700, -900)
tim_delay(200) //задержка приводов на определенное время
    srv_moveptpg5(2, 10, -300, -770, 510, 0)
    srv_moveptpg5(3, 10, -300, -770, 510, 0)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_speed_all(8000) //Задаем скорость всем сервоприводам
return //Конец процедуры


d23_sitdown: //Название (начало) процедуры
    srv_moveptpg5(3, 0,300,-760,620,0) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg5(2, 0,300,-760,620,0)
    srv_moveptpg3(4, -200,-700,0)
    srv_moveptpg3(1, -200,-700,0)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он выпрямлял ноги по очереди в позе «казачок».

2. Запрограммируйте робота так, чтобы он встал на мостик.

Урок 17. Изучение и выполнение программ: работа с базой подпрограмм. Движение рукой «подойди»

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнения 1, 2 и 3. Результаты покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнения роботом движения рукой «подойди».

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

//Главная программа
main:
    gosub standard_pose
    gosub come_here
stop
    goto main  //переход в начало цикла
//Стандартная поза
standard_pose:
```

```
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


come_here: //Название (начало) процедуры
    srv_move(4, -400) //Перемещение заданного привода в заданное положение (№ привода, позиция)
    srv_moveptpg3(4, -200, -700, 900) //Перемещаем группу сервоприводов в заданные позиции
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_moveptpg3(4, -900, -700, 900)
    srv_waitmove_all()
    srv_moveptpg3(4, -200, -700, 900)
    srv_waitmove_all()
    srv_moveptpg3(4, -900, -700, 900)
    srv_waitmove_all()
    srv_moveptpg3(4, -200, -700, 900)
    srv_waitmove_all()
    srv_moveptpg3(4, -900, -700, 900)
    srv_waitmove_all()
    srv_moveptpg3(4, -200, -700, 900)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    gosub standard_pose //Ставим робота в стандартную позу
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он выполнил движение рукой «подойди» (правой рукой).

2. Запрограммируйте робота так, чтобы он выполнил движение рукой «подойди» (левой рукой).

3. Запрограммируйте робота так, чтобы он выполнил взмах руками.

Урок 18. Изучение и выполнение программ: работа с базой подпрограмм. Подъем со спины из положения лежа

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнение 1 и покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнения подъема со спины роботом.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)


//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу


//Главная программа
main:
    gosub standard_pose
    gosub backward_standup
stop
    goto main  //переход в начало цикла


standard_pose: //Стандартная поза
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий return //Конец процедуры
```

```
backward_standup: //Название (начало) процедуры
    srv_speed_all(5500) //Задаем скорость всем сервоприводам
    srv_move_all_ptp(-900,300,0,0,0,0,0,0,150,0,-900,0,-900,300,0,0,0,0,0,0,150,0,-900,0) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
    srv_move_all_ptp(-900,300,-800,0,0,0,0,0,400,-170,-900,0,-900,300,-800,0,0,0,0,0,400,-170,-900,0)
    srv_waitmove_all()
//stop
    srv_move_all_ptp(-100,-700,-800,0,0,0,0,0,-500,-500,260,0,-100,-700,- 800,0,0,0,0,0,-500,-500,260,0)
    srv_waitmove_all()
    srv_move_all_ptp(-50,-800,-700,0,0,0,0,0,-850,-300,650,0,-50,-800,- 700,0,0,0,0,0,-850,-300,650,0)
    srv_waitmove_all()
    srv_move_all_ptp(-500,-300,100,0,0,0,0,0,0,-600,650,0,-500,- 300,100,0,0,0,0,0,0,-600,650,0)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
    srv_speed_all(7000) //Задаем скорость всем сервоприводам
    gosub standard_pose //Ставим робота в стандартную позу
 return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он встал из положения лежа.

Урок 19. Изучение и выполнение программ: работа с базой подпрограмм. Ласточка

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.

2. Самостоятельная работа. Выполните упражнения 1, 2 и 3. Результаты покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнения роботом движения «ласточка».

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

main: //Главная программа
    gosub standard_pose
    gosub wing_move stop
    goto main //переход в начало цикла

standard_pose: //Стандартная поза
   srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
```

```
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


wing_move: //Название (начало) процедуры
    srv_speed_all(1500) //Задаем скорость всем сервоприводам
    srv_move_all_ptp(-200,-600,0,0,0,0,0,120,-90,520,-290,150,-200,-600,0,0,0,0,0,-80,-70,450,-240,-120) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_speed_all(2300)
    srv_move_all_ptp(500,500,0,0,0,0,0,150,150,50,-20,100,500,500,0,0,0,0,0,-70,-20,450,-260,-160)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_move_all_ptp(500,500,0,0,0,0,0,150,50,-640,210,100,500,500,0,0,0,0,0,-70,380,460,-400,-160)
    srv_waitmove_all() srv_move_all_ptp(100,100,450,0,0,0,0,150,-360,50,-20,100,100,100,450,0,0,0,0,-70,600,600,-500,-160)
    srv_waitmove_all() srv_speed_all(5500)
    srv_moveptpg3(1, -200,-200,450) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg3(4, -200,-200,450)
    srv_waitmove_all()
    srv_moveptpg3(1, 200,200,450)
    srv_moveptpg3(4, 200,200,450)
    srv_waitmove_all()
    srv_speed_all(6500)
    srv_moveptpg3(1, -200,-200,450)
    srv_moveptpg3(4, -200,-200,450)
    srv_waitmove_all()
    srv_moveptpg3(1, 200,200,450)
    srv_moveptpg3(4, 200,200,450)
    srv_waitmove_all()
    srv_speed_all(7500)
    srv_moveptpg3(1, -200,-200,450)
    srv_moveptpg3(4, -200,-200,450)
```

```
    srv_waitmove_all()
    srv_moveptpg3(1, 200,200,450)
    srv_moveptpg3(4, 200,200,450)
    srv_waitmove_all()
    srv_speed_all(8500)
    srv_moveptpg3(1, -200,-200,450)
    srv_moveptpg3(4, -200,-200,450)
    srv_waitmove_all()
    srv_moveptpg3(1, 200,200,450)
    srv_moveptpg3(4, 200,200,450)
    srv_waitmove_all()
    srv_speed_all(9500)
    srv_moveptpg3(1, -200,-200,450)
    srv_moveptpg3(4, -200,-200,450)
    srv_waitmove_all()
    srv_moveptpg3(1, 200,200,450)
    srv_moveptpg3(4, 200,200,450)
    srv_waitmove_all()
    srv_speed_all(10500)
    srv_moveptpg3(1, -200,-200,450)
    srv_moveptpg3(4, -200,-200,450)
    srv_waitmove_all()
    srv_moveptpg3(1, 200,200,450)
    srv_moveptpg3(4, 200,200,450) //Перемещаем группу сервоприводов в заданные позиции
    srv_waitmove_all()


tim_delay(1000) //Задержка приводов на определенное время
    srv_speed_all(2300)
    srv_move_all_ptp(800,600,0,0,0,0,0,150,-360,50,-20,100,800,600,0,0,0,0,0,-70,600,600,-
500,-160)
    srv_waitmove_all()
    srv_move_all_ptp(500,500,0,0,0,0,0,150,50,-640,210,100,500,500,0,0,0,0,0,- 70,380,460,-
400,-160)
```

```
    srv_waitmove_all()
    srv_speed_all(1500) //Задаем скорость всем сервоприводам
    srv_move_all_ptp(500,500,0,0,0,0,0,150,150,50,-20,100,500,500,0,0,0,0,0,0,-70,-20,450,-260,
-160)
    srv_waitmove_all()
    srv_move_all_ptp(-200,-600,0,0,0,0,0,120,-90,520,-290,150,-200,-600,0,0,0,0,0,0,-80,-70,450,
-240,-120) //Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all()//Ожидания завершения предшествующих действий
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он повернулся направо и сделал «ласточку». И так по кругу, т. е. 4 раза.

2. Запрограммируйте робота так, чтобы он выполнил «ласточку», потом шаг вперед и поклон.

Урок 20. Изучение и выполнение программ: работа с базой подпрограмм. Кувырок вперед

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.
2. Самостоятельная работа. Выполните упражнение 1 и покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнения кувырка вперед роботом.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

main: //Главная программа
    gosub standard_pose
    gosub down_front stop
    goto main //переход в начало цикла

standard_pose: //Стандартная поза
   srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
```

```
 srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


down_front: //Название (начало) процедуры
    srv_speed_all(8000) //Задаем скорость всем сервоприводам
    gosub sitdown_pose //Ставим робота в стандартную позу
    gosub s_fall_forward
    gosub standard_pose
    srv_speed_all(10000) //Задаем скорость всем сервоприводам
    srv_move_all_ptp(-200,-700,800,0,0,0,0,10,400,450,510,0,-200,-
700,800,0,0,0,0,10,400,450,510,0) //Перемещение всех приводов в заданные позиции с одно-
временным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
    srv_speed_all(12000)
    srv_move_all_ptp(0,-850,850,0,0,0,0,10,-850,0,-450,0,0,-850,850,0,0,0,0,10,-850,0,-450,0)
    srv_waitmove_all()


tim_delay(50) //Задержка приводов на определенное время
    srv_speed_all(5500)
    srv_move_all_ptp(-900,300,-800,0,0,0,0,0,400,-170,-900,0,-900,300,- 800,0,0,0,0,0,400,-
170,-900,0)
    srv_waitmove_all()
    gosub backward_standup
    srv_waitmove_all( ) //Ожидания завершения предшествующих действий
    srv_speed_all(7000) //Задаем скорость всем сервоприводам
    gosub standard_pose //Ставим робота в стандартную позу
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он выполнил кувырок назад.

Урок 21. Изучение и выполнение программ: работа с базой подпрограмм. Отжимание

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.
2. Самостоятельная работа. Выполните упражнение 1 и покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнения отжимания роботом.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

main: //Главная программа
    gosub standard_pose
    gosub prisyad stop
    goto main //переход в начало цикла

standard_pose: //Стандартная поза
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
```

```
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return //Конец процедуры


pushup_routine: //Название (начало) процедуры
    srv_speed_all(5000) //Задаем скорость всем сервоприводам
    gosub sitdown_pose
    srv_speed_all(2000)
    srv_moveptpg3(4, -20,-830,800) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg3(1, -90,-800,770)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
srv_moveptpg5(3, 30,600,-740,610,20)
    srv_moveptpg5(2, 20,570,-730,580,10)
    srv_waitmove_all()
    srv_moveptpg3(4, -20,-830,800)
    srv_moveptpg3(1, -90,-800,770)
    srv_waitmove_all()
    srv_moveptpg5(2, 0,480,-370,640,0)
    srv_moveptpg5(3, 20,540,-400,630,0)
    srv_waitmove_all()
    srv_moveptpg5(3, 50,-260,570,-390,0)
    srv_moveptpg5(2, -10,-320,670,-390,0)
    srv_waitmove_all()
    srv_speed_all(8000)


p_up_st: //Название (начало) процедуры
    v = ir2_read()
    if v != 0 then
end
else end
    if v = 0x0013 then // key square
    gosub pushup
    goto p_up_st
```

```
    goto p_up_end
    goto p_up_st


p_up_end:
    srv_speed_all(8000) //Задаем скорость всем сервоприводам srv_moveptpg3(1, -800, 500, 800)
    srv_moveptpg3(4, -800, 500, 800)
    srv_waitmove_all()
    gosub forward_standup
return //Конец процедуры


pushup: //Название (начало) процедуры
    srv_moveptpg5(3, 50,-260,570,-390,0)
    srv_moveptpg3(4, -890,-110,800)
    srv_moveptpg3(1, -890,-160,770)
    srv_moveptpg5(2, -20,-320,660,-390,0)
    srv_waitmove_all() //Ожидания завершения предшествующих действий
    srv_moveptpg3(4, -20,-830,810) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg3(1, -80,-780,770)
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он выполнил отжимание на одной руке.

Урок 22. Изучение и выполнение программ: работа с базой подпрограмм. Удар вправо

План

1. Экспериментальная часть. Наберите готовую программу и запустите ее.
2. Самостоятельная работа. Выполните упражнение 1 и покажите учителю.

Экспериментируй

В данной лабораторной работе мы освоим практическое применение команд, в том числе индивидуальное управление отдельными сервоприводами, которое будет показано на примере выполнения удара вправо роботом.

Пример кода:

```
//Инициализация приводов
    srv_init(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)

//Подключаем необходимые модули
    include «servo_init.bas»
    include «std_proc.bas»

//Включаем все сервоприводы
    gosub servo_group //дает указания контроллеру, какие приводы в какие группы определять
    gosub servo_on //включает и выключает сервоприводы
    gosub standard_pose //Ставим робота в стандартную позу

main: //Главная программа
    gosub standard_pose
    gosub right_attack stop
    goto main //переход в начало цикла

standard_pose: //Стандартная поза
    srv_move_all_ptp(-200,-700,0,0,0,0,0,0,-70,450,-240,0,-200,-700,0,0,0,0,0,0,-70,450,-240,0)
//Перемещение всех приводов в заданные позиции с одновременным завершением движения
    srv_waitmove_all() //Ожидания завершения предшествующих действий
return//Конец процедуры
```

```
right_attack: //Название (начало) процедуры
    srv_speed_all(2000) //Задаем скорость всем сервоприводам
    srv_moveptpg5(3, 0, 40, 80, -20, -120) //Перемещаем группу сервоприводов в заданные позиции
    srv_moveptpg5(2, 0, -135, 450, -240, 163)
    srv_moveptpg3(4, -200, -700, 520)
    srv_moveptpg3(1, 300, -700, 0)
    srv_move(4, -600) //Перемещение заданного привода в заданное положение (№ привода, позиция)
    gosub wait_g4 srv_speed_all(9000)
    srv_moveg3(1, 740, 700, 0,)
    srv_moveg3(4, 40, -32, 900)
    srv_waitmove_all()//Ожидания завершения предшествующих действий
    srv_speed(4,2500)
    srv_move(4, 0) //Перемещение заданного привода в заданное положение (№ привода, позиция)

tim_delay(500) //Задержка приводов на определенное время
    srv_speed_all(3000)
    gosub standard_pose //Ставим робота в стандартную позу
    srv_speed_all(3500) //Задаем скорость всем сервоприводам
return //Конец процедуры
```

Выполни самостоятельно

1. Запрограммируйте робота так, чтобы он выполнил удар влево.

СПРАВОЧНЫЙ МАТЕРИАЛ

Специфические команды в среде AR-Basic Studio

- **srv_move(mn, val)**

Движение сервомотора с номером mn (с 1 по 24) в позицию val (~ от –1000 до 1000) с учетом текущей скорости.

Пример:

srv_move(1, 500)

Двигает сервомотор, подключенный к первому порту (обычно левая рука робота), в позицию 500.

- **srv_move_all(v1,v2,v3, … , v22, v23, v24)**

Движение всех сервомоторов в заданные позиции (с v1 по v24)

Пример:

srv_move_all(500, 200, 100, 300, 400, … , 600)

Двигает первый сервомотор в позицию 500, второй – в позицию 200, третий – в 100 и т. д. с учетом текущей скорости каждого привода. Для задания общей скорости для всех приводов смотрите команду: srv_speed_all().

- **srv_movefast(mn, val)**

Быстрое перемещение сервомотора с номером mn в позицию val.

Текущая выставленная скорость не учитывается.

Пример:

srv_movefast(1, 500)

Быстрое движение первого мотора в позицию 500.

- **srv_setgroup(mn, g)**

Присвоить сервомотор с номером mn группе g. Данная команда используется для группирования отдельных частей тела робота. Например, группировка сервоприводов паука. Подробнее смотрите в конфигурационном файле моторов – motors.ini. Если вы не уверены в том, что делаете, не меняйте значения в файле.

Если у вас стандартное подключение и конфигурация моторов, то данную команду можете не прописывать вручную для каждого мотора. Создавайте новые программы из шаблона, там по умолчанию написана процедура инициализации групп и моторов.

Пример:

srv_setgroup(1, 1)

Присвоить первый сервомотор группе номер 1.

- **srv_getgroup(mn)**

Команда возвращает номер группы, к которой присвоен мотор mn.

Пример:

a = srv_getgroup(1)

Переменной *a* будет присвоен номер группы, в которой находится первый мотор.

- **srv_moveg6(g, v1, v2, v3, v4, v5, v6)**

Перемещение одновременно 6 сервомоторов группы g в позиции v1, v2, … , v6.

Пример:

srv_moveg6(1, 500, 200, 300, 100, 400, 600)

Первые шесть моторов в группе 1 подвинутся в положения 500, 200, 300, 100, 400, 600 в соответствии с их порядковым номером. То есть если в первой группе находятся моторы 1, 15, 7, 4, 13, 3, то 1 подвинется в позицию 500, 3 – в 200, 4 – в 300, 7 – в 100, 13 – в 400 и 15 – в 600.

- **srv_moveg8(g, v1, v2, v3, v4, v5, v6, v7, v8)**

Перемещение 8 сервомоторов группы g в позиции v1, v2, …, v8.

Принцип тот же, что и в команде srv_moveg6.

- **srv_moveg2, srv_moveg3, srv_moveg4, srv_moveg5, srv_moveg7, srv_moveg9, srv_moveg10, srv_moveg11, srv_moveg12**

Групповое перемещение сервомоторов. См. описание команды srv_moveg6. Отличие только в количестве параметров.

- **srv_moveptp(mn, val)**

PTP расшифровывается как point-to-point. Перемещение мотора с номером mn в позицию val таким образом, чтобы все сервомашинки в той же группе, что и mn, закончили движение одновременно.

Пример:

srv_moveptp(1, 500)

Перемещает сервопривод с номером 1 в позицию 500 с такой скоростью, чтобы он закончил перемещение синхронно с остальными приводами в его группе. Если в данный момент нет движения других приводов в этой группе, то скорость движения будет та, которая была указана для данного сервомотора или для всех.

- **srv_moveptpg6(g, v1, v2, v3, v4, v5, v6)**

Перемещение группы из 6 сервомоторов в заданные позиции, аналогично srv_moveg6, причем все моторы закончат движение одновременно. То есть приводы двигаются с разной, рассчитанной скоростью.

Пример:

srv_moveptpg6(1, 500, 200, 300, 100, 600, 800)

Приводы в первой группе в соответствии с их порядковым номером по возрастанию двигаются в позиции 500, 200, 300, …, причем каждый привод с разной скоростью – такой, чтобы все приводы закончили движение одновременно.

Пример:

srv_moveptpg6(1, 200, 300, 400, 500, 600, 700)

Перемещение сервомоторов в первой группе в позиции 200, 300, 400, … в соответствии с их порядковым номером.

Причем так, чтобы все моторы закончили перемещение одновременно.

- **srv_moveptpg3, srv_moveptpg5, srv_moveptpg8**

См. команду srv_moveptpg6. Отличается только количество параметров. Для робота MR-200 используется в основном только групповое движение трех моторов (srv_moveptpg3).

- **moveg(g, x)**

Перемещение всех сервомоторов в группе g в позицию x.

Пример:

moveg(1, 500)

Перемещение всех сервомоторов в первой группе в позицию 500.

- **srv_waitmove(mn)**

Ожидание того момента, когда сервомотор mn закончит перемещение.

- **srv_waitmove_all()**

Ожидание того момента, когда все сервоприводы закончат перемещение.

- **srv_enable(mn, state)**

Установка сервопривода mn в состояние state: 0 – отключен, 1– включен, 2 – цифровой вход, 3 – цифровой выход.

Цифровой вход и выход используются для работы с иными подключенными устройствами (не сервоприводами), такими как диоды, выключатели и пр. Если инициализировать порты в состояние цифрового входа или выхода, функциями port_in() и port_out() можно считывать и записывать значения с таких портов.

Пример:

srv_enable(1, 1)

Включает первый сервопривод. Данная команда обычно используется в начале программы для включения всех активных приводов. Чтобы не писать каждый раз, можно воспользоваться любым из шаблонов, где описана процедура инициализации.

- **srv_enable_all(s1, s2, s3, … , s22, s23, s24)**

Установка состояния для всех приводов. См. команду srv_enable().

Пример:

srv_enable_all(1, 1, 1, 0, … , 1, 1, 1)

Установит состояние «включен» для приводов 1, 2, 3, … и состояние «отключен» для привода 4 и любых других, где указано состояние 0.

- **srv_getpos(mn)**

Возвращает позицию мотора с номером mn.

Пример:

a = srv_getpos(1)

Переменной a будет присвоена та позиция, в которой находится на данный момент первый привод.

- **srv_getdiff(mn, v)**

Возвращает признак достижения сервомотором заданной позиции en.

Пример:

a = srv_getdiff(1, 200)

Переменной a будет присвоено значение 0, если сервомотор достиг позиции 200, или 1, если не достиг.

- **srv_speed(mn, val)**

Задать скорость val для одного мотора. От 0 до 15 000 (относительные единицы).

Пример:

srv_speed(1, 1000)

Задает первому мотору скорость 1000.

- **srv_speed_all(val)**

Задать скорость для всех сервомоторов.

Пример:

srv_speed_all(1000)

Задает всем моторам скорость 1000.

- **srv_speedg(g, val)**

Задать скорость для всех машинок в группе g.

Пример:

srv_speedg(1, 1500)

Задает всем моторам, принадлежащим к группе 1, скорость 1500.

- **port_in(id)**

Получить значение с цифрового порта с номером id. Данный порт должен быть инициализирован как цифровой порт. См. команду srv_enable().

Пример:

a = port_in(5)

Переменной a будет присвоено значение, считанное с пятого цифрового порта.

- **port_out(id, code)**

Послать значение code на цифровой порт с номером id. Порт должен быть инициализирован как цифровой порт. См. команду srv_enable().

Пример:

port_out(5, 10)

Пошлет значение 10 в пятый цифровой порт на контроллере.

- **peek(addr)**

Прочитать ячейку дополнительной памяти размером 2k × 4.

Пример:

a = peek(1)

Прочитает значение первой дополнительной ячейки памяти.

- **poke(addr, data)**

Записать data в ячейку дополнительной памяти.

Пример:

poke(1, 30)

Запишет число 30 в первую ячейку дополнительной памяти.

- **ir_read()**

Прочитать код с инфракрасного порта.

Пример:

a = ir_read()

Запишет в переменную a код, полученный с ИК-порта.

- **rf_read()**

Прочитать код с радиоканала

- **include «filename.bas»**

Подключает дополнительный модуль к исходному файлу.

Все дополнительные модули хранятся в файле include, куда можно добавлять собственные. Из файлов такого типа подключаются не сами команды в файле, а только отдельные процедуры, которые далее можно будет вызывать из исходной программы.

Пример:

include «servo_init.bas».

Подключит дополнительный модуль инициализации сервоприводов. В нем хранятся такие процедуры, как servo_on, servo_group, servo_off и т. д. Далее эти процедуры можно вызывать с помощью команды gosub.

Литература

1. Введение в программирование на AR-Basic Studio. Инструкция. – Магнитогорск, 2007.

2. Глушков С.В. Программирование на языке Visual Basic 6.0 / С.В. Глушков, А.С. Сурядный. – М.: ООО «АСТ», Фолио, 2003.

3. Ловин Д. Создаем робота-андроида своими руками / Д. Ловин. – М.: ДМК Пресс, 2007.

4. Учебно-методический комплекс с лабораторными роботами серии AR-1001M: руководство пользователя // ЗАО «Андроидные роботы» – М., 2009.

5. Учебно-методический комплекс с лабораторными роботами серии AR-1001M: руководство по программированию на языке RoboBasic // ЗАО «Андроидные роботы». – М., 2009.

6. Сысойкина М. Русский андроид: будущее начинается / М. Сысойкина // Мир ПК. – 2009. – № 1.

Электронное учебное издание

сетевого распространения

Шарафеева Ландыш Рамилевна

Гуськов Вадим Сергеевич

Сазонов Евгений Валерьевич

ПРОГРАММИРОВАНИЕ АНТРОПОМОРФНЫХ РОБОТОВ

Учебное пособие

Корректор ***А.Н. Егорова***

Компьютерная верстка ***Р.М. Абдрахмановой***

Подписано к использованию 24.11.2021
Формат 60x84 1/16. Гарнитура «Times New Roman».
Усл. печ. л. 4,3. Заказ 163/11.

Издательство Казанского университета

420008, г. Казань, ул. Профессора Нужина, 1/37
тел. (843) 233-73-59, 233-73-28