Кульминский Д. Д.
Косарев А. А.
Григоров М. Ю.

Методика программирования промышленных роботов-манипуляторов на примере ABB IRB1600

Учебно-методическое пособие



Университет «Сириус»

Научный центр информационных технологий
и искусственного интеллекта

Д. Д. Кульминский, А. А. Косарев, М. Ю. Григоров

МЕТОДИКА ПРОГРАММИРОВАНИЯ ПРОМЫШЛЕННЫХ РОБОТОВ-МАНИПУЛЯТОРОВ НА ПРИМЕРЕ ABB IRB1600

Учебно-методическое пособие

Электронное издание
локального распространения

Санкт-Петербург
Наукоемкие технологии
2026

© Кульминский Д. Д., Косарев А. А.,
Григоров М. Ю., 2026
ISBN 978-5-00271-176-5

УДК 007.52:004.896
ББК 32.816
К90

Рецензенты:
В. А. Ненашев, доктор технических наук;
А. С. Штин, кандидат технических наук

К90 Кульминский Д. Д., Косарев А. А., Григоров М. Ю. Методика программирования промышленных роботов-манипуляторов на примере ABB IRB1600 [Электронный ресурс]: учебно-методическое пособие / Д. Д. Кульминский, А. А. Косарев, М. Ю. Григоров. – Электрон, текстовые дан. (7,1 Мб). – СПб.: Наукоемкие технологии, 2026. – 88 с. – 1 электрон., опт. диск (CD-ROM).

ISBN 978-5-00271-176-5

Учебно-методическое пособие посвящено практическому освоению методов программирования промышленного робота-манипулятора ABB IRB 1600 в среде ABB RobotStudio с последующим переносом управляющей программы на реальный контроллер. В издании последовательно изложены теоретические сведения о среде RobotStudio и языке RAPID, системах координат (база, инструмент, рабочий объект), типах движений. Пособие содержит подробный пошаговый порядок выполнения работы: построение виртуальной модели робототехнической ячейки, создание и запуск виртуального контроллера, ручное управление и задание позиции, формирование рабочего объекта, задание целевых точек и траекторий для движения, синхронизацию виртуальной ячейки с кодом RAPID, запуск симуляции с визуализацией траектории инструмента и перенос программы на реального робота. Также приведены разделы, посвящённые технике безопасности при работе с промышленным роботом, работе с пультом FlexPendant, калибровке рабочего объекта и инструмента на реальном оборудовании, возможностям языка RAPID.

Учебно-методическое пособие ориентировано на студентов всех форм обучения, проходящих подготовку по направлениям 09.03.01 и 15.04.06, изучающих промышленную робототехнику, автоматизацию технологических процессов и программирование роботов-манипуляторов. Издание может быть полезно преподавателям, инженерам-технологам и специалистам, занимающимся внедрением и отладкой робототехнических комплексов на базе оборудования ABB.

© Кульминский Д. Д., Косарев А. А.,
Григоров М. Ю., 2026

ISBN 978-5-00271-176-5

Учебное издание

Кульминский Данил Дмитриевич
Косарев Артем Андреевич
Григоров Максим Юрьевич

Методика программирования промышленных роботов-манипуляторов на примере ABB IRB1600

Учебно-методическое пособие

Электронное издание
локального распространения

Минимальные системные требования:

- процессор: Intel x86, x64, AMD x86, x64 не менее 1 ГГц;
- оперативная память RAM ОЗУ: не менее 512 МБайт;
- свободное место на жестком диске (HDD): не менее 120 МБайт;
- операционная система: Windows XP и выше;
- Adobe Acrobat Reader;
- дисковод CD-ROM;
- мышь.

Издается в авторской редакции
Публикуется из оригинал-макета, подготовленного авторами
Главный редактор *В. М. Коровин*

Издательство «Наукоемкие технологии»
ООО «Корпорация «Интел Групп»
https://publishing.intelgr.com
E-mail: publishing@intelgr.com
Тел.: +7 (812) 945-50-63

Подписано к использованию 18.09.2026 г.
Объем издания – 7,1 Мб.
Комплектация издания – 1 CD.
Тираж 100 CD.

ISBN 978-5-00271-176-5



Содержание

Введение **6**
1. Теоретические сведения **10**
1.1. Промышленный робот-манипулятор ABB IRB 1600 10
1.2. Среда ABB RobotStudio и язык RAPID 10
1.3. Системы координат: база, инструмент и рабочий объект 11
1.4. Типы движений: MoveJ и MoveL 11
1.5. Параметры движения: скорость и зоны сглаживания 12
1.6. Представление ориентации: кватернионы и конфигурация осей 12
1.7. Подпружинённый карандаш и калибровка 13
2. Порядок выполнения работы **14**
2.1. Интерфейс RobotStudio: навигация и горячие клавиши 14
2.2. Построение сцены: робот, окружение и подпружинённый карандаш 14
2.3. Создание и запуск виртуального контроллера 20
2.4. Ручное управление и задание домашней позиции 23
2.5. Рабочий объект стола: система координат по трём точкам 27
2.6. Создание рабочего объекта через диалог Create Workobject 31
2.7. Задание целевых точек траектории 36
2.8. Формирование путей движения и выбор типа интерполяции 37
2.9. Синхронизация с кодом RAPID и точка входа main 41
2.10. Запуск симуляции и визуализация траектории (TCP Trace) 44
2.11. Сглаживание траектории зонами 48
2.12. Структура кода RAPID: robtarget, кватернионы и конфигурация 50
2.13. Сохранение программы для последующего переноса на робота 52
3. Техника безопасности при работе с промышленным роботом **55**
3.1. Общие положения 55
3.2. Аварийные ситуации 56
3.3. Режимы работы и зоны безопасности 57
3.4. Специфика работы с подпружиненным карандашом 58
3.5. Действия после завершения работы 59
4. Работа с пультом FlexPendant и перенос программы на робота **60**
4.1. Интерфейс FlexPendant 60
4.2. Загрузка программы с USB-носителя 61
4.3. Калибровка рабочего объекта (wobjdata) на реальном столе 63

4.4. Калибровка инструмента (tooldata) 65
4.5. Первый запуск программы на реальном роботе 66
4.6. Возможные ошибки при переносе 68
5. Возможности языка RAPID 70
5.1. Типы данных переменных и констант 73
5.2. Операторы и инструкции 75
5.3. Массивы 79
5.4. Обработчики прерывания и ошибок 80
6. Требования к отчету и критерии оценивания 85
Глоссарий основных терминов 86
Контрольные вопросы 87
Список литературы 88

Введение

Современное автоматизированное производство немыслимо без применения промышленных роботов-манипуляторов. Они обеспечивают высокую производительность, стабильное качество продукции, возможность работы в опасных и вредных условиях, а также гибкость переналадки при смене номенклатуры изделий. Одним из ключевых факторов эффективного использования робототехники является качество управляющих программ, которые определяют точность, скорость и надёжность выполнения технологических операций.

Особое место среди задач промышленной робототехники занимают контурные операции, к которым относится сварка, резка, покраска, нанесение клеевых швов, а также написание текста или нанесение маркировки. Такие операции предъявляют повышенные требования к точности позиционирования инструмента, плавности траектории и корректности ориентации в пространстве. Ошибки, допущенные на этапе программирования, могут привести к браку продукции, повреждению инструмента или оборудования, а в худшем случае — к аварийной ситуации.

Настоящее учебно-методическое пособие посвящено освоению методики алгоритмизации и программирования промышленного робота-манипулятора ABB IRB 1600 для обеспечения точности и качества в автоматизированных производствах. В качестве прикладной задачи выбрано написание подпружиненным карандашом букв или инициалов участников команды на листе бумаги, закреплённом на рабочем столе. Несмотря на кажущуюся простоту, эта задача требует от обучающегося комплексного применения знаний и навыков: построения виртуальной модели рабочей ячейки, задания пользовательской системы координат, выбора типов движений, настройки скорости и зон сглаживания, анализа структуры данных, а также переноса программы на реальное оборудование с учётом неизбежных погрешностей.

Актуальность пособия обусловлена несколькими факторами. Во-первых, промышленные роботы ABB широко распространены в российской и мировой промышленности, а среда RobotStudio является стандартным инструментом автономного программирования и моделирования. Во-вторых, переход от симуляции к реальному оборудованию неизбежно сопровождается

расхождениями, связанными с неточностью калибровки, нелинейностью реального мира и ограничениями кинематики. Понимание этих процессов и умение их компенсировать — важнейшая компетенция современного инженера. В-третьих, существующие учебные материалы зачастую ограничиваются описанием интерфейса или простейших движений, тогда как данное пособие предлагает сквозной цикл: от теоретических основ до запуска программы на реальном роботе.

Цель пособия — сформировать у обучающихся устойчивые практические навыки автономного программирования промышленного робота-манипулятора ABB IRB 1600 в среде RobotStudio и на языке RAPID, а также подготовить их к самостоятельному решению прикладных задач контурного движения с обеспечением требуемой точности и качества.

Для достижения поставленной цели решаются следующие задачи:

— освоение среды ABB RobotStudio и языка RAPID;

— понимание систем координат (база, инструмент, рабочий объект) и их роли в программировании;

— освоение типов движений MoveJ, MoveL, MoveC и параметров скорости и зон сглаживания;

— изучение представления ориентации через кватернионы и вектор конфигурации;

— построение виртуальной модели роботизированной рабочей ячейки и создание виртуального контроллера;

— формирование рабочего объекта стола методом трёх точек;

— задание целевых точек и путей движения, синхронизация с кодом RAPID;

— запуск симуляции с визуализацией траектории TCP Trace и сглаживание зон;

— перенос программы на реальный контроллер, калибровка инструмента и рабочего объекта;

— анализ результатов, выявление и устранение типичных ошибок.

Структура пособия построена по принципу последовательного усложнения материала. Первый раздел содержит теоретические сведения о роботе, среде RobotStudio, системах координат, типах движений, параметрах движения, представлении ориентации и подпружиненном карандаше. Второй раздел описывает порядок выполнения работы — от настройки интерфейса и построения сцены до запуска симуляции и переноса программы на реального робота. Третий раздел охватывают технику безопасности. Четвертый раздел посвящен работе с пультом FlexPendant — от запуска управляющей программы до калибровки рабочего объекта и инструмента. Пятый раздел раскрывает возможности языка RAPID — описаны типы данных, логические операторы, условные операторы, циклы ветвления, процедуры - обработчики прерывания. В шестом разделе приведены требования к отчёту по структуре IMRAD, являющейся типовой для подготовки научно-технологической документации. Также методическое пособие имеет глоссарий и список литературы.

Адресация. Пособие предназначено для обучающихся инженерных и технических направлений подготовки, изучающих промышленную робототехнику, автоматизацию технологических процессов и программирование роботов-манипуляторов. Издание может быть полезно преподавателям, инженерам-технологам и специалистам, занимающимся внедрением и отладкой робототехнических комплексов на базе оборудования ABB.

Методические рекомендации по работе с пособием. Материал следует изучать последовательно, не пропуская теоретические разделы, поскольку каждый последующий этап опирается на предыдущий. Перед выполнением практической части необходимо ознакомиться с техникой безопасности. При работе в RobotStudio рекомендуется сохранять промежуточные версии проекта, чтобы иметь возможность вернуться к предыдущему состоянию. При переносе программы на реального робота следует строго соблюдать «ручной» режим и пониженную скорость. Все возникающие ошибки рекомендуется фиксировать и анализировать — это способствует более глубокому пониманию процессов, происходящих в системе.

Авторы надеются, что данное пособие поможет обучающимся не только освоить конкретные инструменты и приёмы программирования робота ABB IRB 1600, но и сформировать инженерное мышление, необходимое для решения более сложных задач автоматизации производства.

1. Теоретические сведения

1.1. Промышленный робот-манипулятор ABB IRB 1600

Промышленный робот-манипулятор представляет собой автоматическую перепрограммируемую машину, предназначенную для перемещения инструмента или объекта манипулирования по заданной пространственной траектории. Робот ABB IRB 1600, используемый в настоящей работе, является шарнирным (антропоморфным) манипулятором с шестью вращательными степенями подвижности, что обеспечивает позиционирование инструмента в произвольной точке рабочей зоны с произвольной ориентацией.

В лабораторном комплексе применяется модификация с грузоподъёмностью 10 кг и максимальным вылетом руки 1,45 м (обозначение IRB 1600-10/1.45). Выбор именно этой модели в проекте обусловлен тем, что физически такой робот установлен в лаборатории, и разрабатываемая программа в дальнейшем будет проверяться и исполняться на реальном оборудовании. Высокая повторяемость позиционирования делает робот этого класса удобным средством для отработки задач контурного движения, к которым относится и написание букв.

1.2. Среда ABB RobotStudio и язык RAPID

RobotStudio — это среда автономного программирования и моделирования, позволяющая разрабатывать и проверять управляющие программы без остановки реального производства. В основе среды лежит технология виртуального контроллера (Virtual Controller), который исполняет тот же системный код RobotWare, что и физический контроллер робота. Благодаря этому поведение цифрового двойника в симуляции с высокой точностью соответствует поведению реальной установки.

Управляющие программы для роботов ABB пишутся на специализированном высокоуровневом языке RAPID. Программа на RAPID состоит из модулей, которые содержат объявления данных (например, целевых точек типа robtarget) и процедуры (PROC) с последовательностями инструкций движения и логики. Среда RobotStudio позволяет формировать траекторию графически — расстановкой точек и путей в сцене — и затем

автоматически синхронизировать её с текстовым кодом RAPID, который можно редактировать вручную.

1.3. Системы координат: база, инструмент и рабочий объект

Корректное задание траектории невозможно без понимания используемых систем координат. В работе задействованы три ключевые системы:

— базовая система координат (Base Frame) — связана с основанием робота и является отсчётной для всех его движений;

— система координат инструмента (Tool Frame), определяющая характеристическую точку инструмента — Tool Center Point (TCP). Именно положение и ориентация TCP программируются в инструкциях движения; в данной работе TCP совмещён с остриём подпружинённого карандаша;

— система координат рабочего объекта (Work Object, wobj), привязанная к обрабатываемой детали или, как в нашем случае, к плоскости стола.

Использование собственного рабочего объекта для стола является не формальностью, а признаком грамотного инженерного подхода. Задание всех точек относительно wobj0 (мировой системы) считается плохой практикой, поскольку при переносе программы на реальную установку точное положение и высота стола относительно робота заранее неизвестны. Если же траектория описана относительно рабочего объекта стола, то при отладке достаточно сместить или повернуть один рабочий объект, и вся траектория автоматически скорректируется, не требуя пересчёта каждой точки.

1.4. Типы движений: MoveJ и MoveL

Язык RAPID предоставляет несколько типов инструкций перемещения, из которых в работе используются две основные:

— MoveJ (Joint Move, движение по сочленениям) — робот перемещается из текущего положения в целевую точку наиболее удобным для приводов образом; траектория TCP в декартовом пространстве при этом не является прямолинейной. Такое движение быстрее и безопаснее при переходе между удалёнными позициями;

— MoveL (Linear Move, линейное движение) — TCP движется строго по прямой линии в декартовом пространстве, что необходимо при вычерчивании отрезков букв.

Подход к первой точке пути и возврат в домашнюю позицию обычно программируют инструкцией MoveJ. Это связано с явлением сингулярности (singularity) — особой конфигурации манипулятора, в которой две оси выстраиваются соосно, число эффективных степеней подвижности падает, а решение обратной задачи кинематики становится неопределённым. Линейное движение через окрестность сингулярности может потребовать недопустимо высоких скоростей в сочленениях; движение MoveJ позволяет роботу обойти такую конфигурацию, придя в стартовую точку удобным для него путём.

1.5. Параметры движения: скорость и зоны сглаживания

Каждая инструкция перемещения содержит аргументы, определяющие динамику движения. Аргумент скорости (например, v100) задаёт линейную скорость TCP в мм/с. Аргумент зоны (zonedata) определяет точность прохождения целевой точки:

— значение fine означает, что робот точно останавливается в целевой точке, прежде чем продолжить движение; это требуется в узловых точках букв, где траектория меняет направление под острым углом;

— значение вида z1, z10, z100 задаёт радиус зоны сглаживания (в миллиметрах), в пределах которой робот срезает угол, не доходя до точки, и плавно переходит к следующему сегменту. Сглаживание повышает скорость и плавность движения, но снижает геометрическую точность в данной точке.

1.6. Представление ориентации: кватернионы и конфигурация осей

Целевая точка в RAPID описывается данными типа robtarget, который включает четыре составляющие: координаты положения TCP [x, y, z] в миллиметрах; ориентацию инструмента, заданную единичным кватернионом [q1, q2, q3, q4]; конфигурацию осей робота и положения внешних осей.

Кватернион используется для компактного и устойчивого (свободного от так называемого «шарнирного замка») представления ориентации в пространстве вместо углов Эйлера. Вектор конфигурации (confdata)

фиксирует, в какой именно из нескольких кинематически эквивалентных конфигураций робот должен прийти в точку, то есть какое из возможных решений обратной задачи кинематики выбрано. Понимание структуры robtarget важно при ручном редактировании кода и при анализе причин расхождения симуляции и реального движения.

1.7. Подпружиненный карандаш и калибровка

В качестве пишущего инструмента в работе применяется карандаш, закреплённый на подпружиненном держателе (compliant tool). Податливость инструмента вдоль вертикальной оси выполняет роль пассивной обратной связи по усилию: даже при наличии погрешностей позиционирования и не идеальности поверхности карандаш сохраняет контакт с бумагой и не ломается, не пробивая стол.

Необходимость такого решения становится очевидной при переносе программы из симуляции на реального робота. Во-первых, при загрузке программы возникают погрешности координат, требующие калибровки рабочего объекта (как правило, корректировки смещения по оси z). Во-вторых, реальный мир вносит нелинейности — шероховатость и не плоскостность стола, прогиб листа и прочие факторы, — которые невозможно полностью учесть в модели.

2. Порядок выполнения работы

В настоящем разделе пошагово описана последовательность действий в среде RobotStudio — от настройки интерфейса до переноса готовой программы на реальный контроллер. Каждый этап сопровождается снимком экрана рабочей сцены.

2.1. Интерфейс RobotStudio: навигация и горячие клавиши

Перед началом построения сцены полезно освоить базовые приёмы навигации в трёхмерном окне. Для управления видом используются следующие сочетания клавиш:

— Ctrl + Shift + ЛКМ — вращение (орбита) сцены вокруг точки обзора;

— Ctrl + ЛКМ — панорамное перемещение сцены.

Базовая система координат манипулятора ориентирована следующим образом: ось X направлена на наблюдателя, ось Z — вертикально вверх, ось Y — слева направо (рис. 1). Понимание ориентации осей необходимо для корректного задания направлений при создании рабочего объекта.

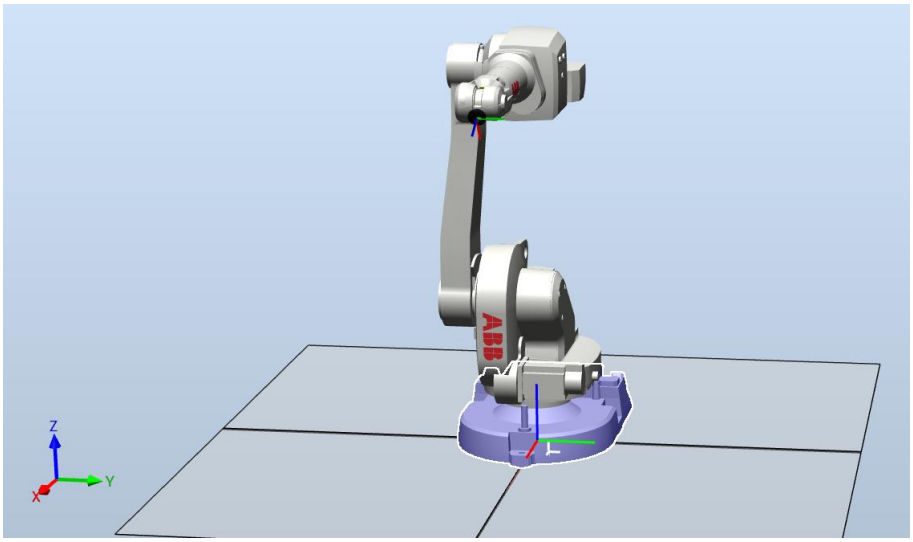


Рисунок 1. Виртуальная модель робототехнической ячейки

2.2. Построение сцены: робот, окружение и подпружиненный карандаш

> *Сцена (станция) в RobotStudio собирается из библиотечных компонентов: модели робота, элементов окружения и инструмента. На данном этапе формируется геометрическая обстановка, максимально соответствующая реальному лабораторному комплексу.*

Для добавления пользовательской библиотеки окружения (подставки, стола, захвата с карандашом и т. п.) импортируется соответствующий компонент через меню добавления библиотек (рис.2-7).

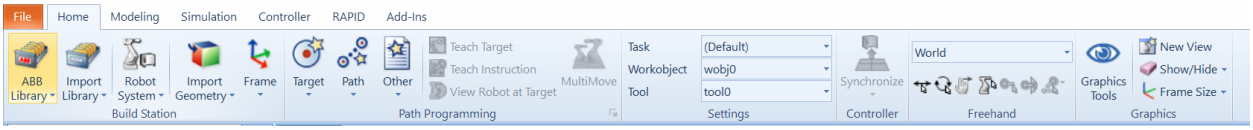


Рисунок 2. Меню добавления компонентов виртуальной модели робототехнической ячейки



Рисунок 3. Окно выбора конкретной модели робота

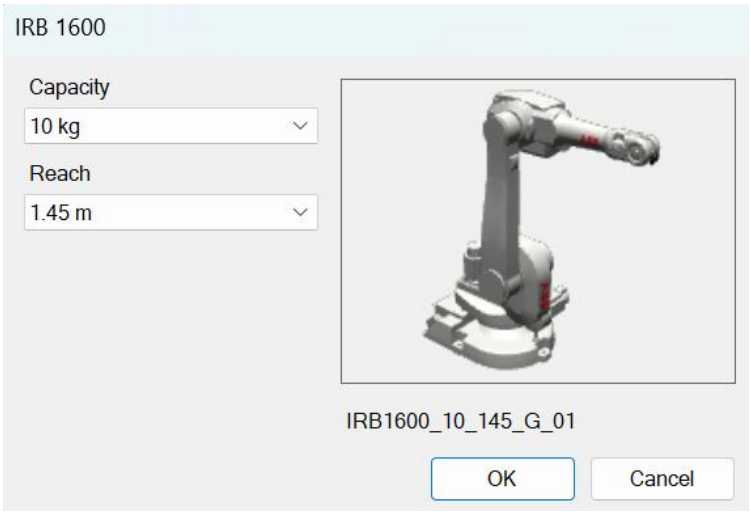


Рисунок 4. Окно выбора параметров конкретной модели робота

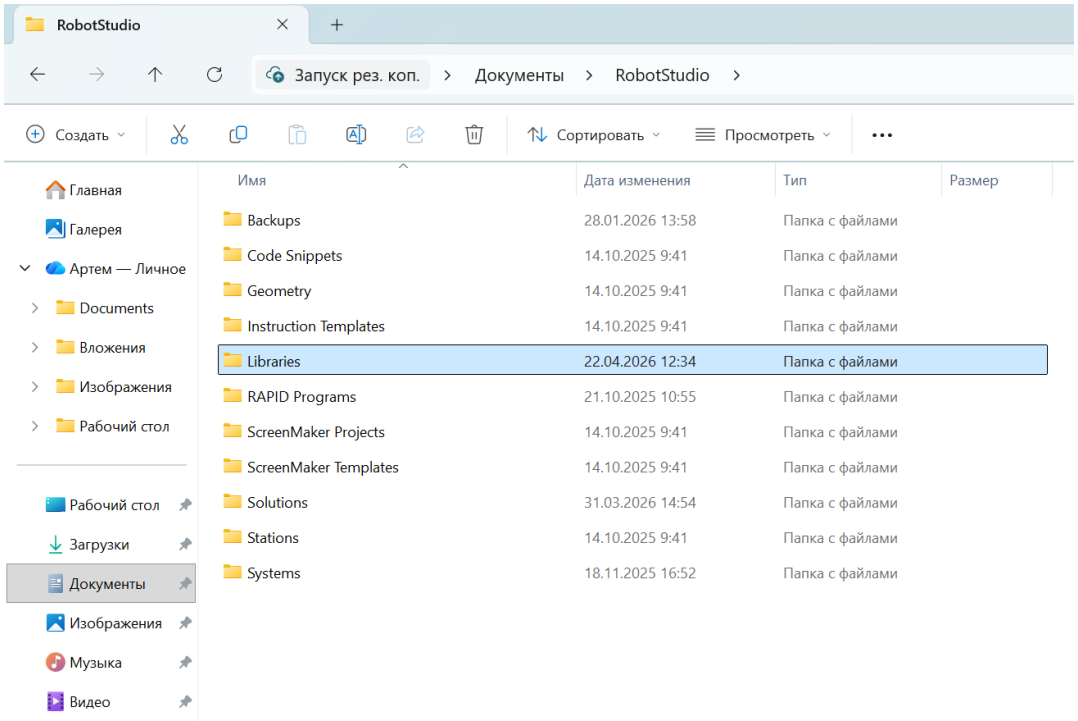


Рисунок 5. Директория на локальном устройстве с файлами RobotStudio

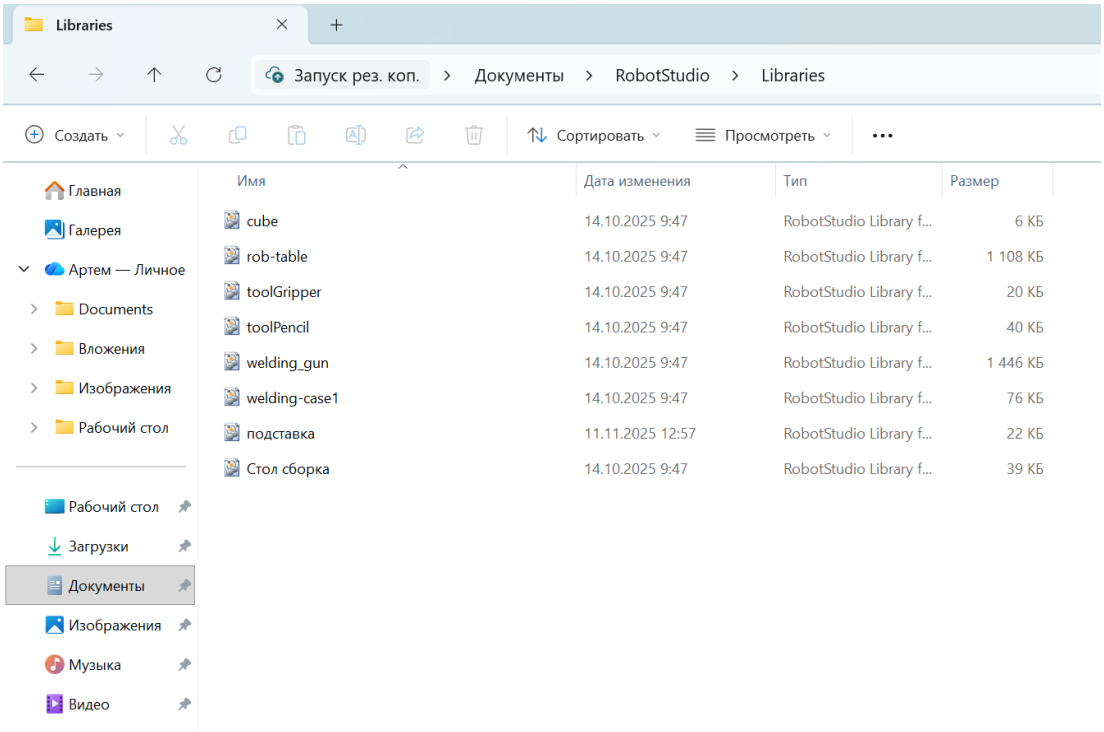


Рисунок 6. Директория на локальном устройстве для размещения необходимых 3D моделей для виртуальной ячейки

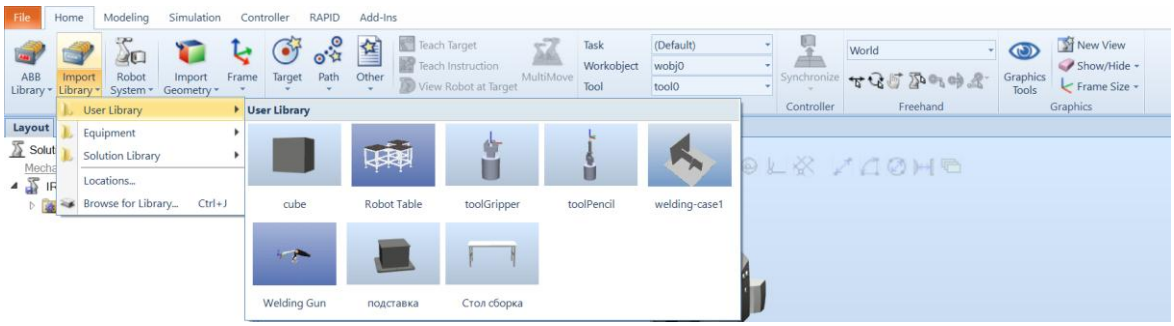


Рисунок 7. Меню добавления компонентов виртуальной модели робототехнической ячейки после загрузки файлов в директории «Libraries»

После добавления подставки она по умолчанию загружается в начало координат — там же, где расположено основание робота (рис. 8). Необходимо корректно установить манипулятор на подставку, согласовав их взаимное положение.

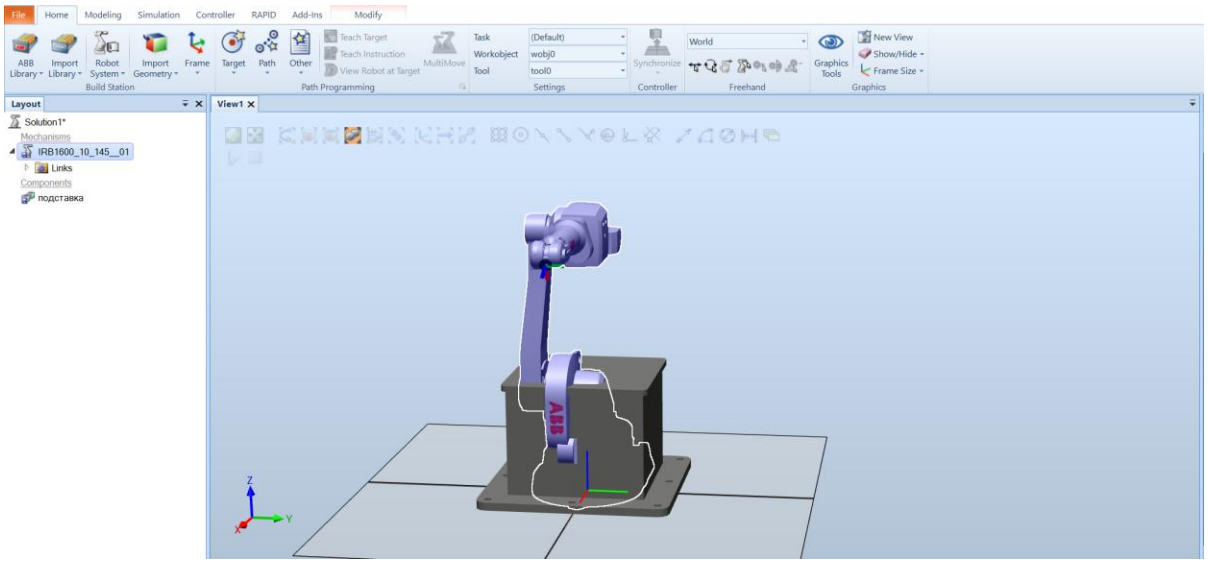


Рисунок 8. Виртуальная модель робототехнической ячейки после добавления компонента «Подставка»

В дереве компонентов вкладки Layout (рис. 9) вызывается контекстное меню манипулятора (правая кнопка мыши) для задания его положения относительно подставки.

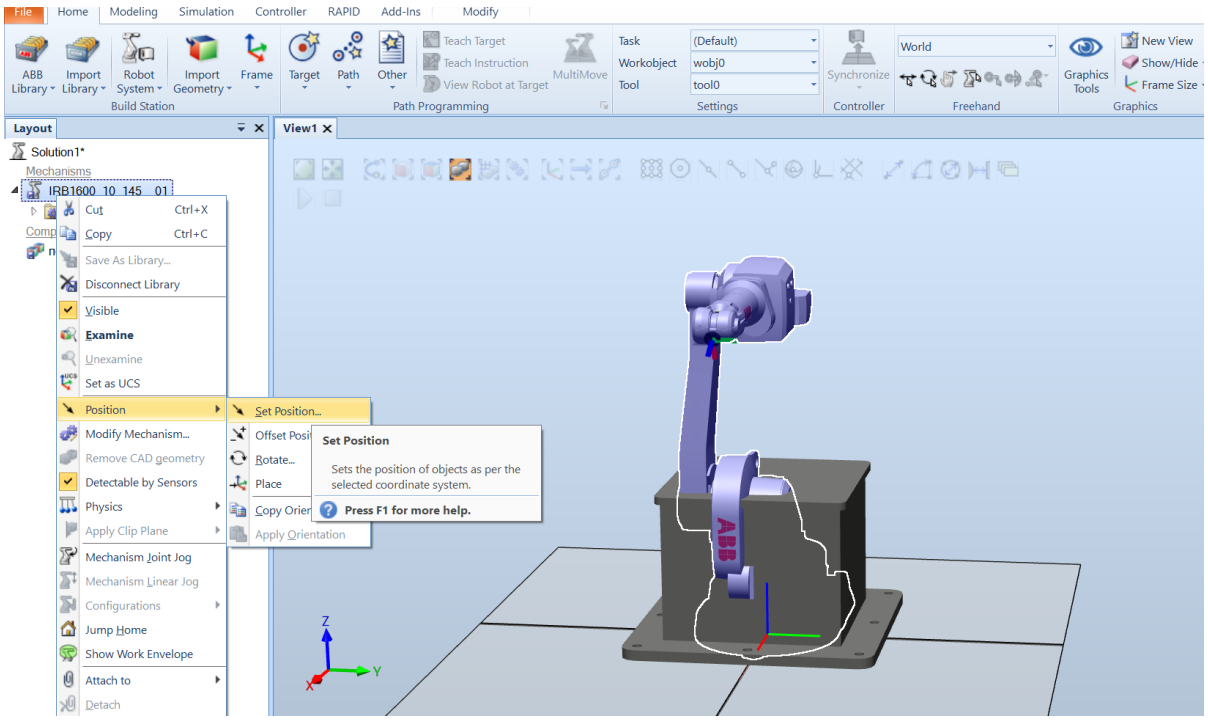


Рисунок 9. Дерево компонентов вкладки «Layout»

Заданное смещение подтверждается нажатием «Apply» (рис. 10).

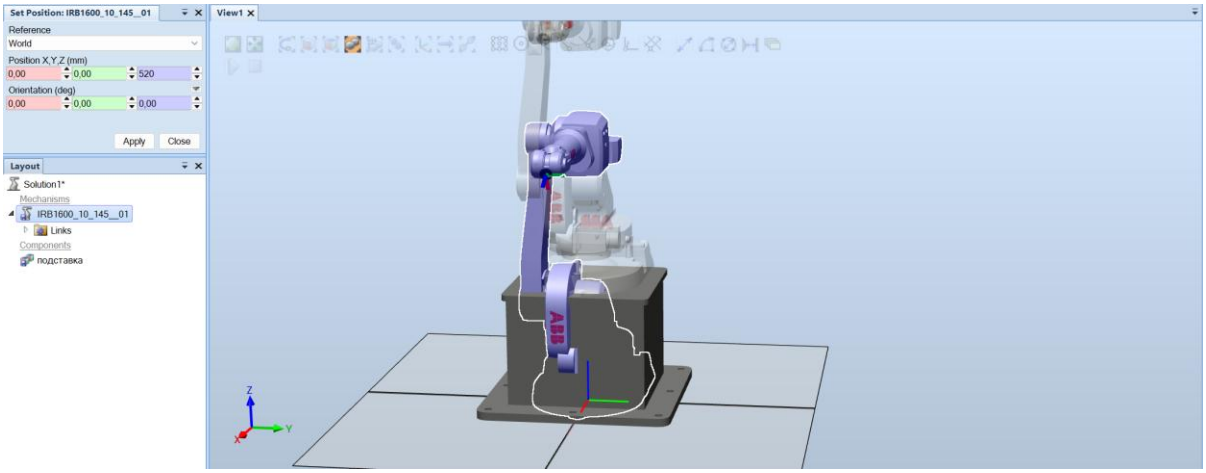


Рисунок 10. Окно для ввода заданного смещения по оси «Z» и результат изменения виртуальной ячейки

Инструмент toolPencil (подпружинённый карандаш, рис. 11) присоединяется к фланцу последнего звена робота IRB 1600-10/1.45 перетаскиванием в дереве компонентов (рис. 12); при этом TCP инструмента связывается с системой координат фланца.

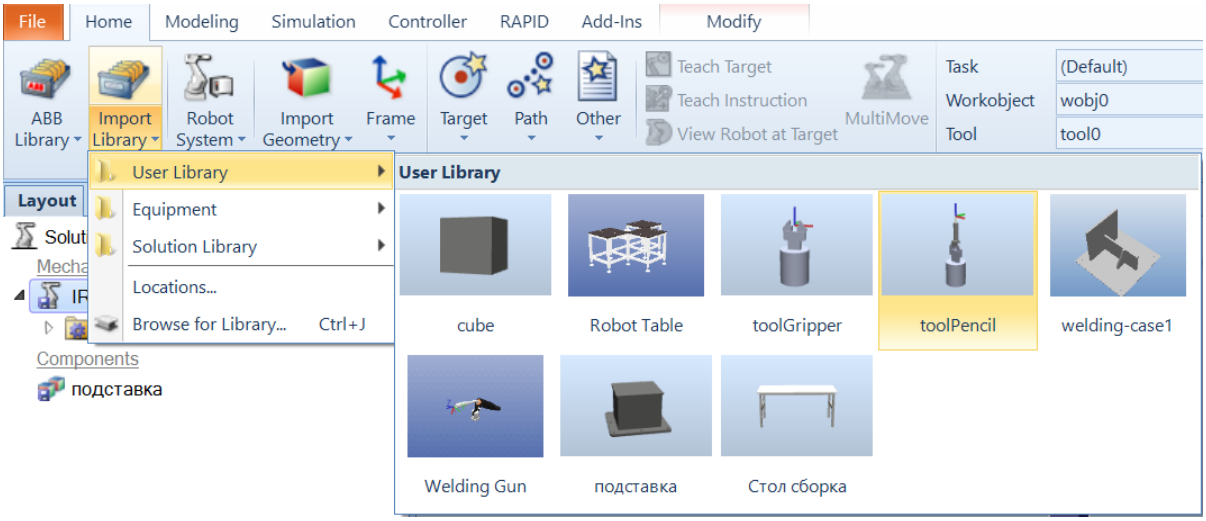


Рисунок 11. Инструмент toolPencil в дереве компонентов

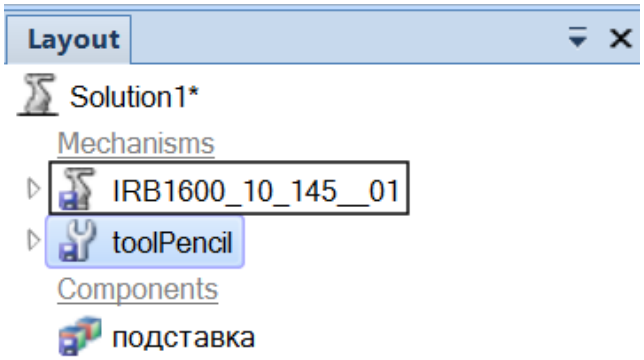


Рисунок 12. Меню дерева компонентов «Layout» для переноса инструмента на робота

Аналогичным образом в проект добавляются стол и подставка, после чего стол при необходимости поворачивается в требуемую ориентацию (рис. 13).

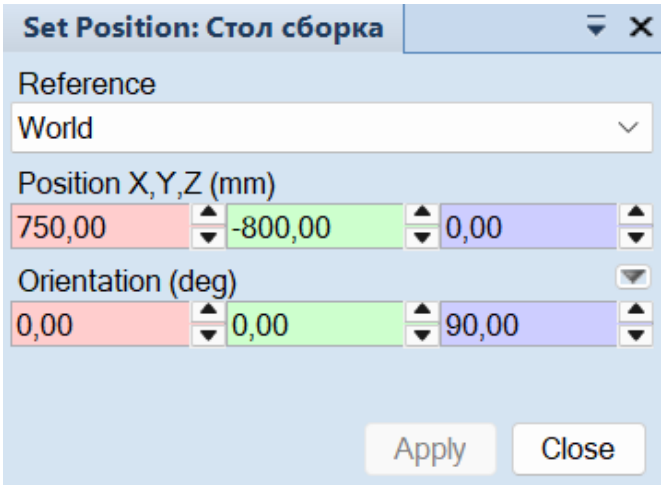


Рисунок 13. Меню для задания смещений компонентов виртуальной модели робототехнической ячейки

2.3. Создание и запуск виртуального контроллера

> *Виртуальный контроллер исполняет системное программное обеспечение RobotWare и обеспечивает поведение цифрового двойника, идентичное реальному роботу. Версию RobotWare следует выбирать совпадающей с версией прошивки контроллера в лаборатории.*

Для дальнейшей работы создаётся виртуальная версия контроллера, соответствующая выбранной модели робота (рис. 14).

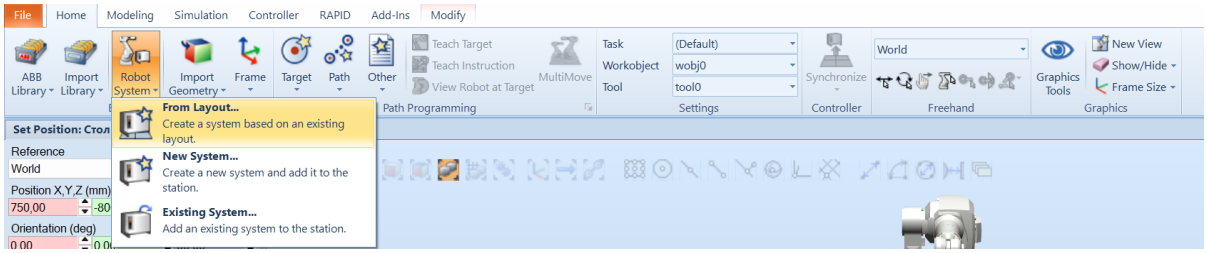


Рисунок 14. Меню выбора виртуального контроллера

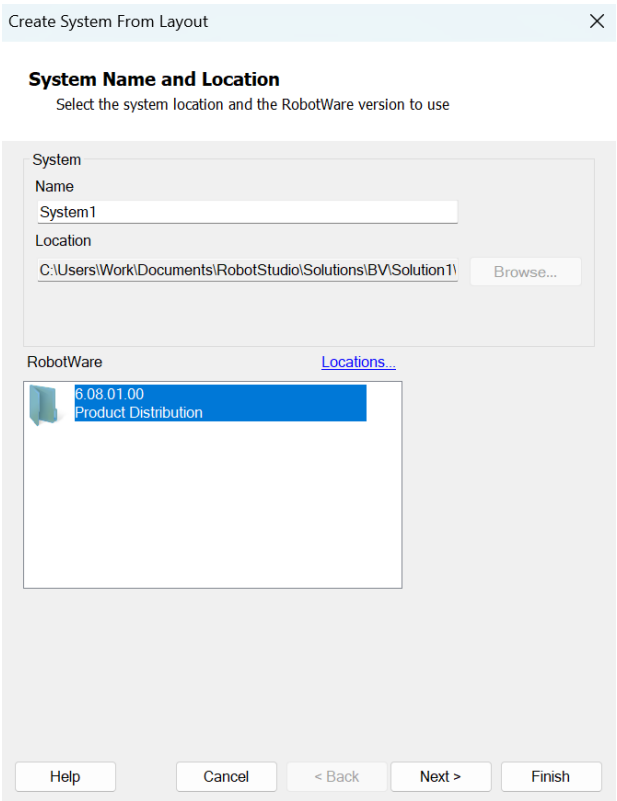


Рисунок 15 Меню мастера создания контроллера

В мастере создания контроллера выполняется переход к следующему шагу нажатием «Next» (рис. 15). После генерации конфигурации создание контроллера завершается нажатием «Finish» (рис. 16).

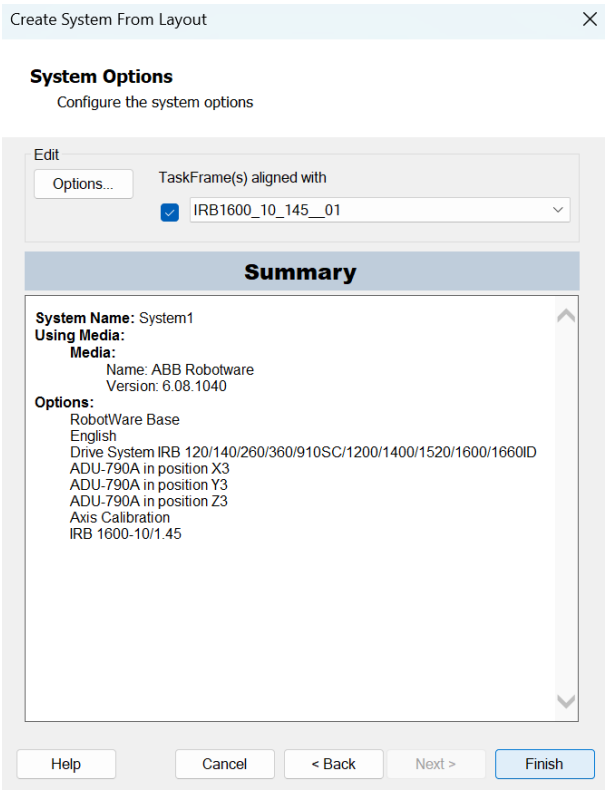


Рисунок 16. Меню мастера создания контроллера с выбранными опциями контроллера робота

Дожидаемся окончания загрузки: индикатор состояния Controller Status должен загореться зелёным цветом, что свидетельствует о готовности контроллера к работе (рис. 17).

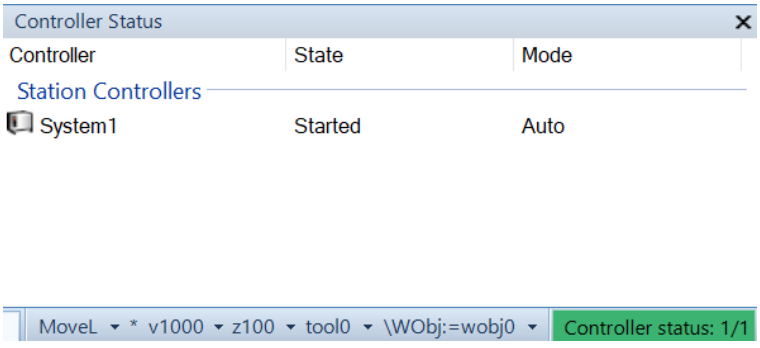


Рисунок 17. Меню статуса контроллера

2.4. Ручное управление и задание домашней позиции

> *Ручное перемещение (Jog) позволяет привести манипулятор в требуемое положение и запомнить его как целевую точку. Различают линейное перемещение TCP (Jog Linear) и пошарнирное перемещение (Jog Joint).*

В качестве активного инструмента выбирается tool, связанный с TCP подпружинённого карандаша (рис. 18).

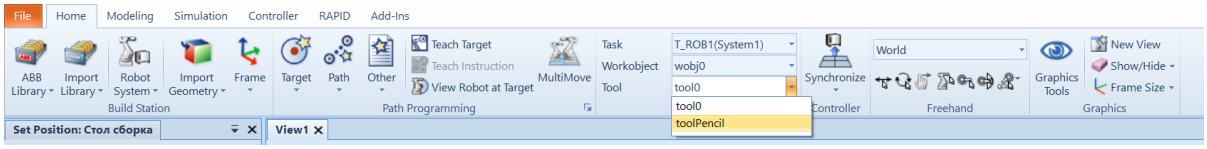


Рисунок 18. Выбор активного инструмента «tool»

Режим Jog Linear обеспечивает линейное перемещение TCP манипулятора (рис. 19) и доступен через вкладку Layout (рис. 20).

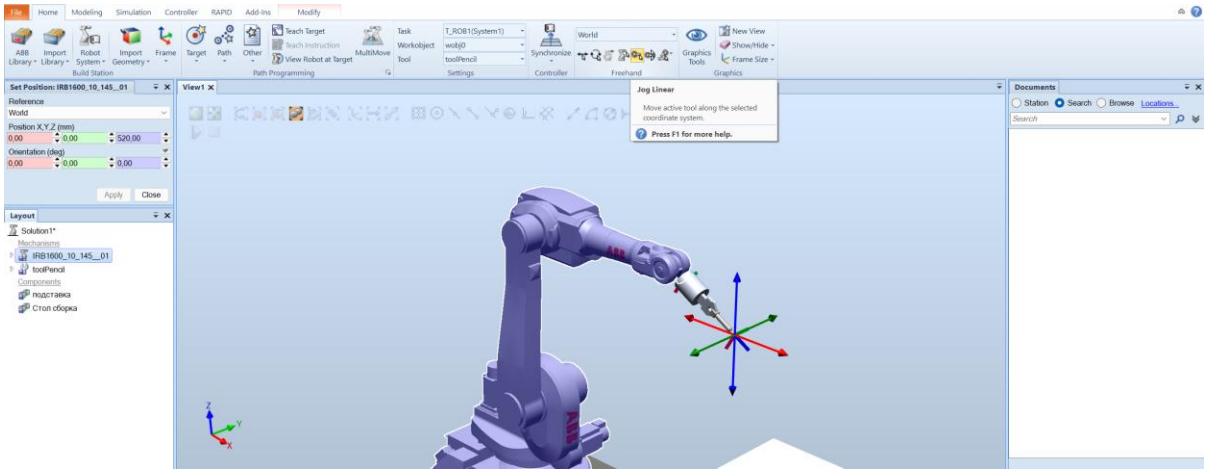


Рисунок 19. Меня выбора режима перемещения «Jog Linear» или «Jog Joint»

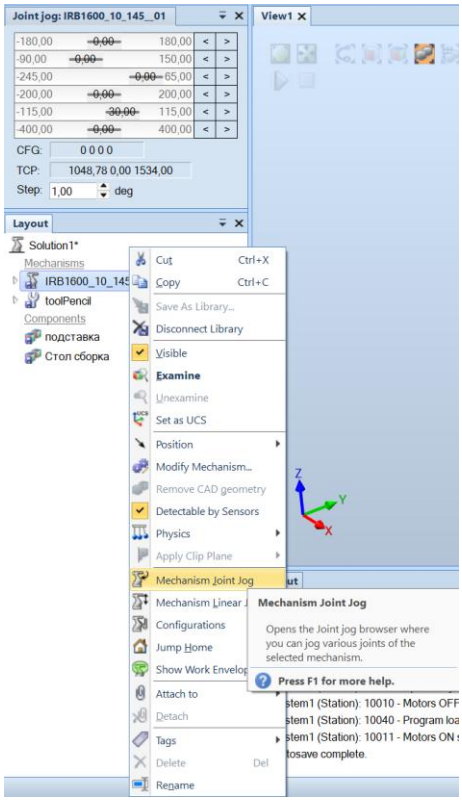


Рисунок 20. Вкладка Layout для выбора режима перемещения

Режим Jog Joint (пошарнирное перемещение) (рис. 21) позволяет независимо управлять каждой осью манипулятора и вызывается через контекстное меню робота.

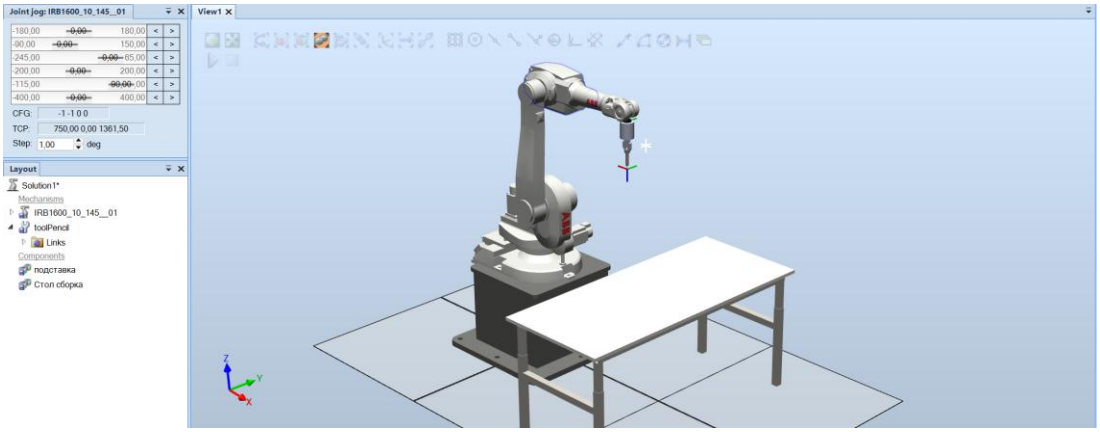


Рисунок 21. Активный режим Jog Joint (пошарнирное перемещение)

Манипулятор приводится в домашнюю позицию: его поднимают над столом так, чтобы между вторым и третьим звеньями образовался прямой угол (рис. 21). Эта конфигурация удобна как исходная и безопасна с точки зрения сингулярностей.

Текущее положение манипулятора сохраняется как целевая точка соответствующей командой запоминания позиции (рис. 22).

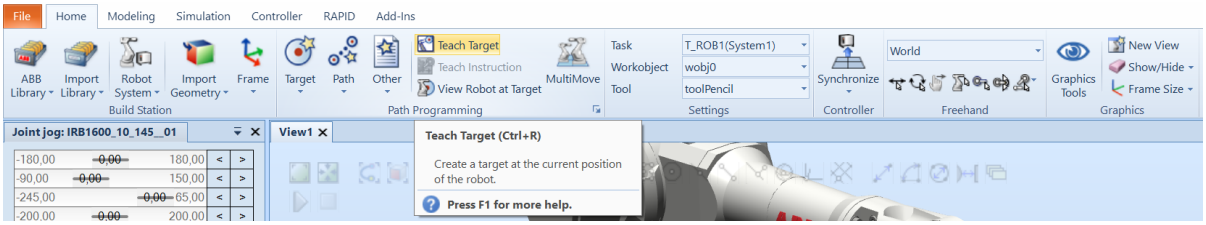


Рисунок 22. Кнопка для сохранения текущего положения манипулятора в качестве целевой точки

Если при добавлении целевой точки возникает предупреждение (рис. 23), его подтверждают нажатием «Yes»; опционально устанавливается флаг «Don’t show message about this again», чтобы не выводить сообщение повторно.

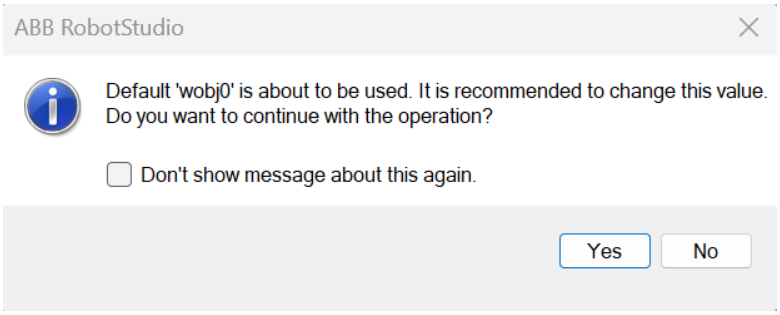


Рисунок 23. Всплывающее окно с предупреждением о том, что целевая точка связана с системой координат wobj0

В случае случайного закрытия окон Layout или Path & Targets их восстанавливают через раздел «Customize Quick Access Toolbar», установив флаги напротив необходимых панелей (рис. 24).

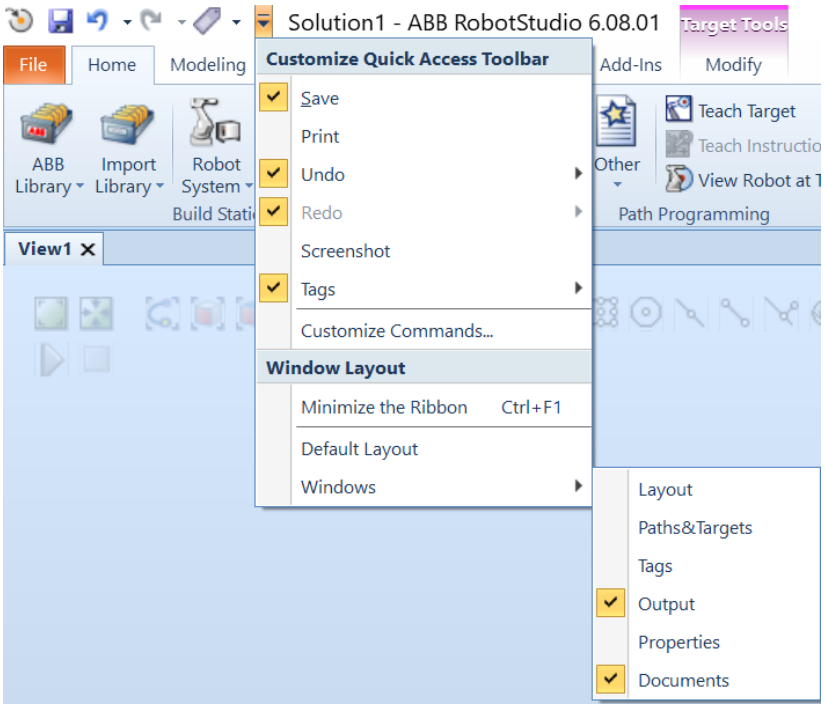


Рисунок 24. Окно «Customize Quick Access Toolbar»

Добавленная точка отображается в разделе «Path & Targets» (рис. 25). Для удобства первую точку целесообразно переименовать в Home.

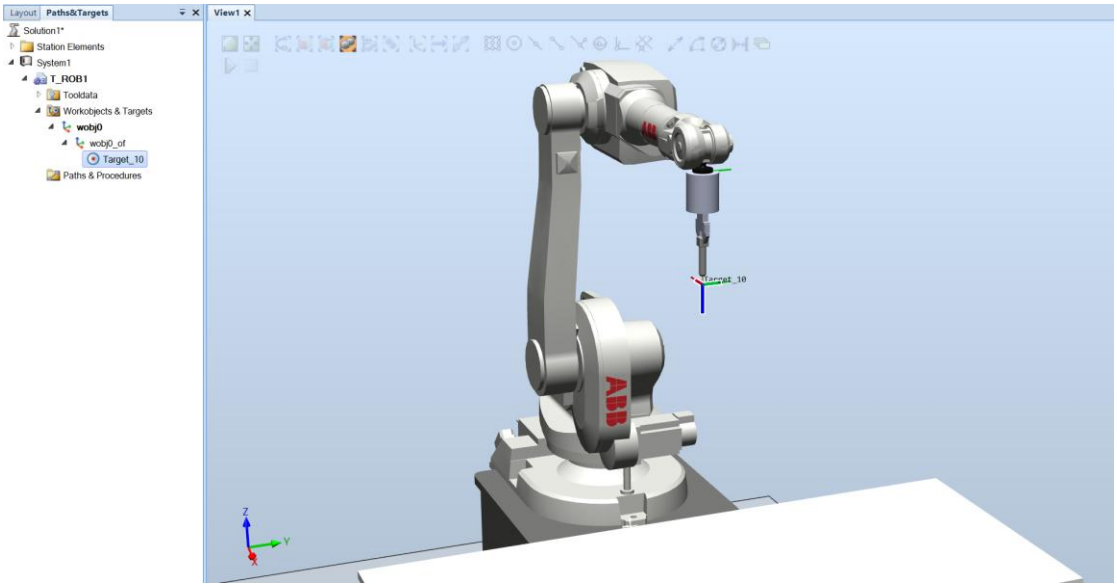


Рисунок 25. Окно «Path & Targets»

Функция «Jump to target» позволяет автоматически привести манипулятор в выбранную целевую позицию (рис. 26).

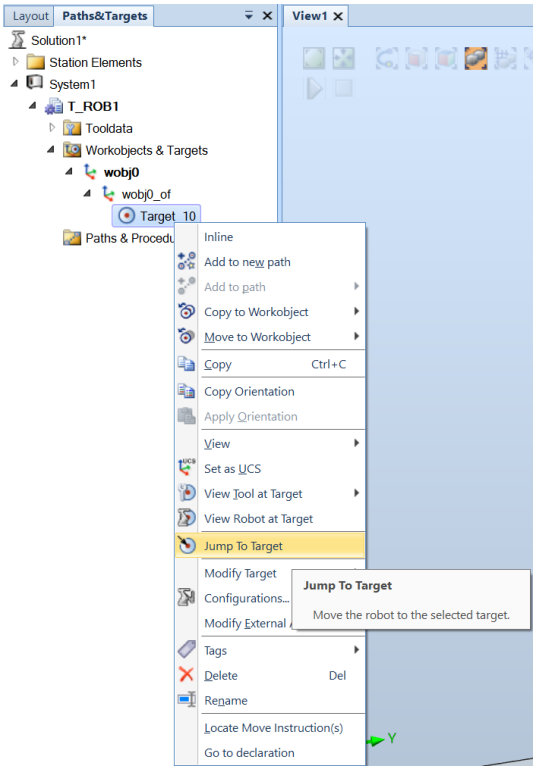


Рисунок 26. Вкладка функции «Jump to target» в меню «Path & Targets»

2.5. Рабочий объект стола: система координат по трём точкам

> *Как отмечено в п. 1.3, все точки траектории следует задавать относительно рабочего объекта, связанного с плоскостью стола. Это позволяет при отладке на реальном оборудовании корректировать положение траектории смещением одного рабочего объекта.*

Для выполнения работы требуется задавать точки букв относительно стола. Если задавать их прежним способом, точки будут принадлежать wobj0, что неудобно и считается плохим тоном. Правильным решением является создание для плоскости стола собственного рабочего объекта (Workobject), которому будут принадлежать все точки траектории (рис. 26).

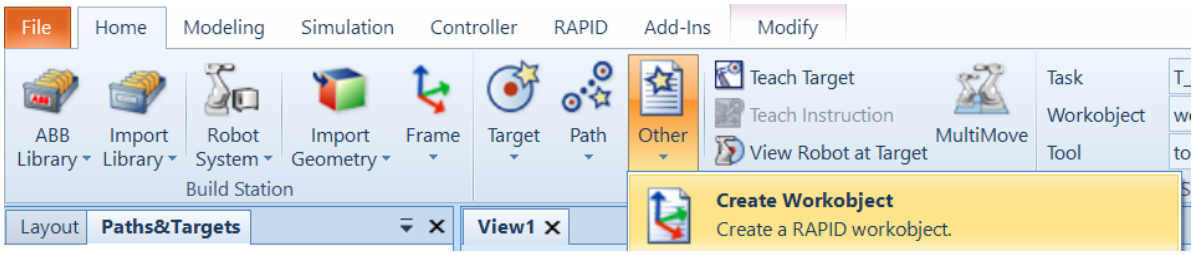


Рисунок 26. Меню создания рабочего объекта (Workobject)

Новый рабочий объект для удобства последующей работы переименовывают, например, в workobject_table.

При создании рабочего объекта задают его оси координат. Для этого применяется метод определения системы координат по трём точкам (рис. 27).

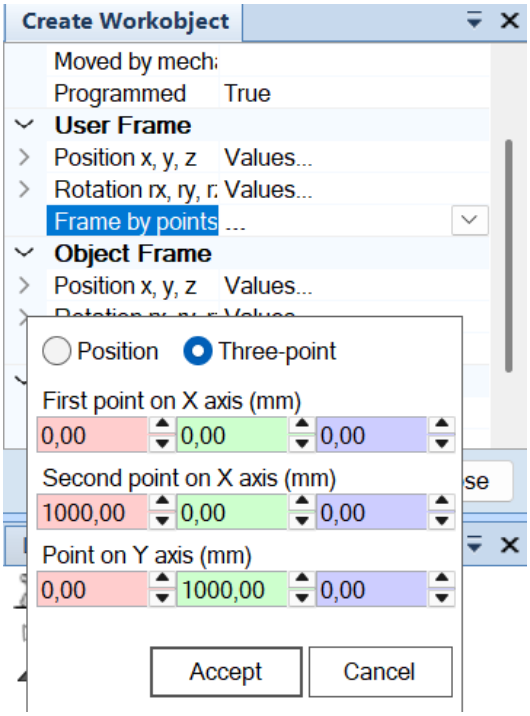


Рисунок 27. Меню редактирования рабочего объекта (Workobject)

Для точной привязки к геометрии стола используют функцию «Snap Object» (рис. 28): поскольку точки задаются на поверхности стола, эта функция выделяет характерные (реперные) точки поверхности, облегчая их выбор.

При выборе трёх точек для системы координат задают две точки вдоль оси X (определяя её положительное направление) и одну точку для оси Y;

остальное RobotStudio рассчитывает автоматически. После выбора точек нажимают Create.

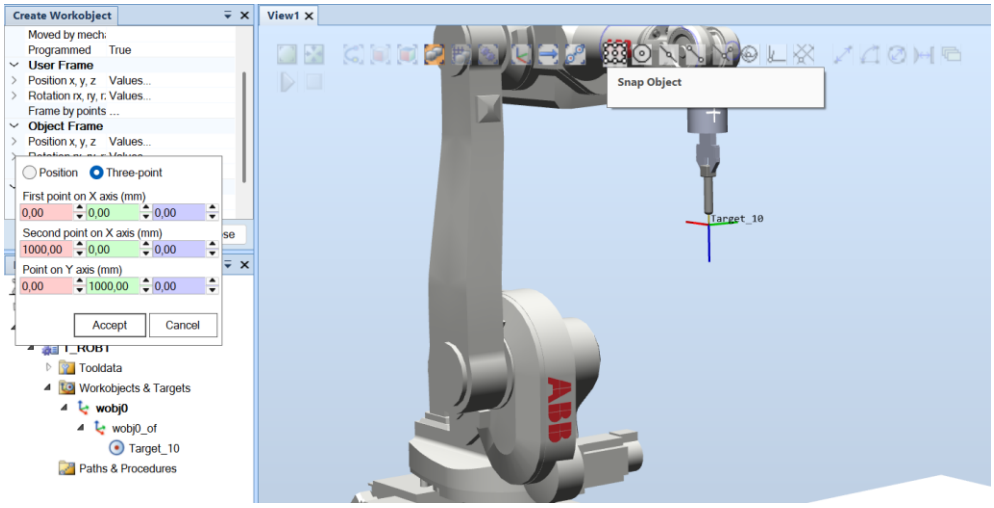


Рисунок 28. Выбор функции «Snap Object»

Систему координат рабочего объекта удобно задавать в углу рабочей поверхности стола (рис. 29).

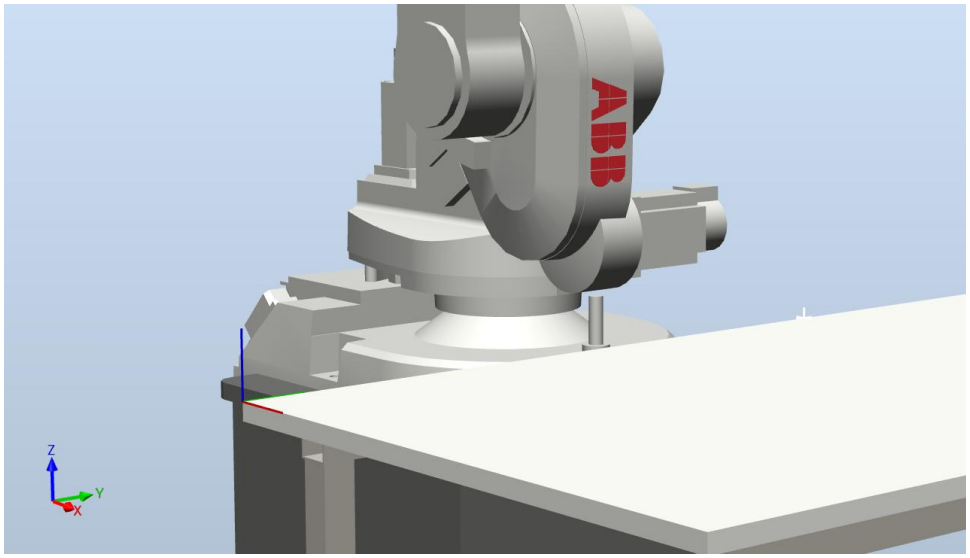


Рисунок 29. Расположение системы координат рабочего объекта в виртуальной модели робототехнической ячейки

После создания автоматически выбирается новый рабочий объект, который теперь отображается в дереве проекта (рис. 30).

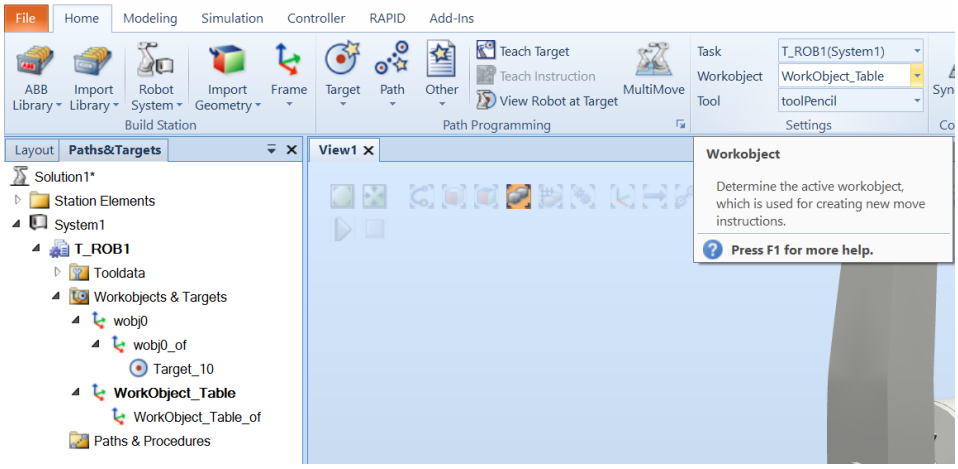


Рисунок 30. Дерево проекта с созданным рабочим объектом

Манипулятор приводят в рабочую область стола и задают новую целевую точку (рис. 31).

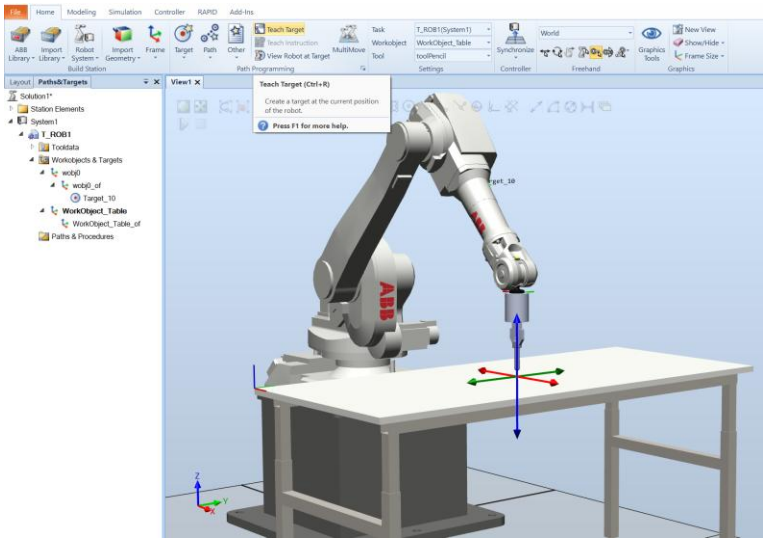


Рисунок 31. Манипулятор в новой целевой точки относительно нового рабочего объекта

Все вновь создаваемые точки теперь добавляются в рабочий объект «WorkObject_Table» (рис. 32).

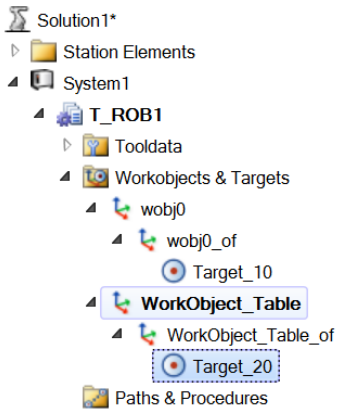


Рисунок 32. Дерево проекта с новым рабочим объектом

2.6. Создание рабочего объекта через диалог Create Workobject

> *Ниже приведён детализированный порядок создания рабочего объекта, привязанного к плоскости стола, с использованием диалогового окна Create Workobject и метода трёх точек.*

Во вкладке дополнительных инструментов выбирают команду «Create Workobject» (рис. 32).

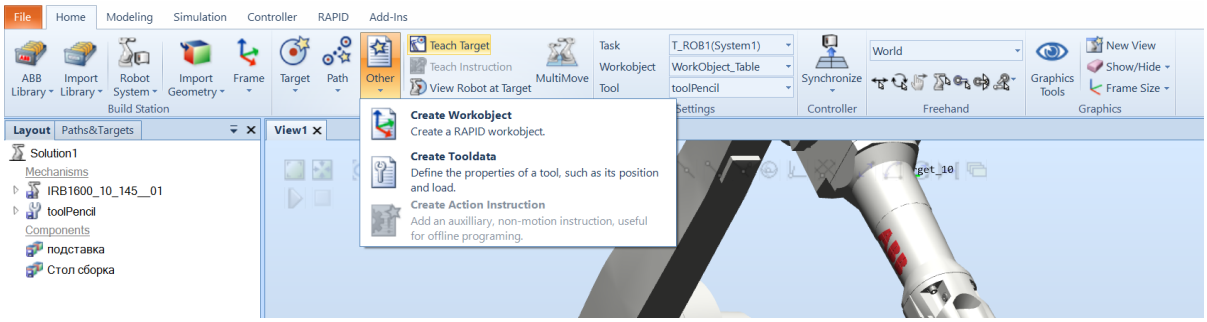


Рисунок 33. Создание системы координат, привязанной к плоскости стола

Слева открывается панель «Workobject» с параметрами создаваемого рабочего объекта (рис. 34). Рабочему объекту присваивают имя «wbj_Table».

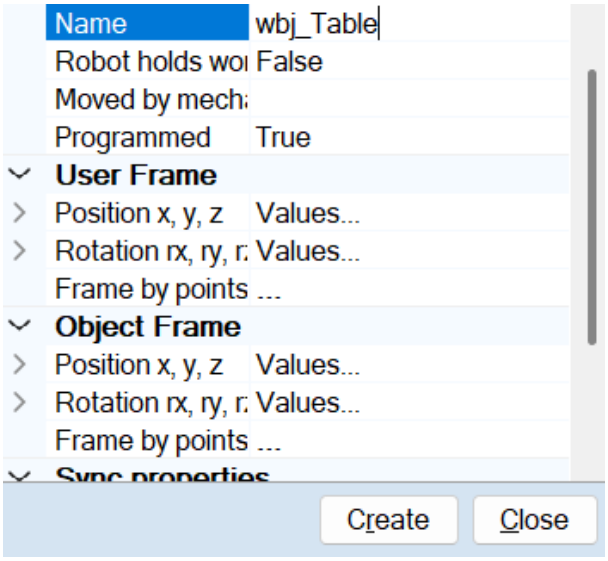


Рисунок 34. Панель для редактирования рабочего объекта «Workobject»

Во вкладке «**User Frame**» (рис. 35) система координат задаётся по трём точкам. Для этого нажимают на стрелку у поля «Frame by points».

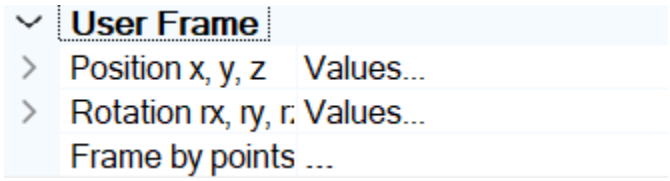


Рисунок 35. Вкладка «User Frame»

Способ задания переключают на «Three point» (по трём точкам), см. рис. 36.

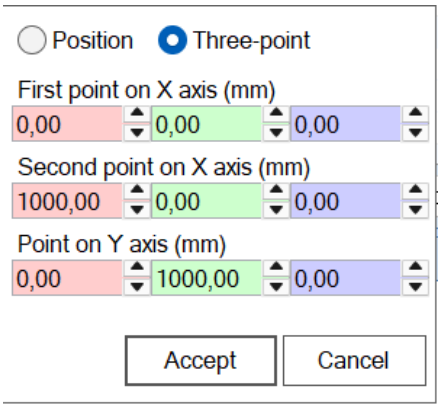


Рисунок 36. Вкладка «Frame by points»

Точки выбирают так, чтобы ось X шла по ширине стола, ось Y — по длине стола вправо, а ось Z — вертикально вверх от поверхности.

Активируют режим привязки «Snap Object» (рис. 37) для точного выбора характерных точек поверхности.

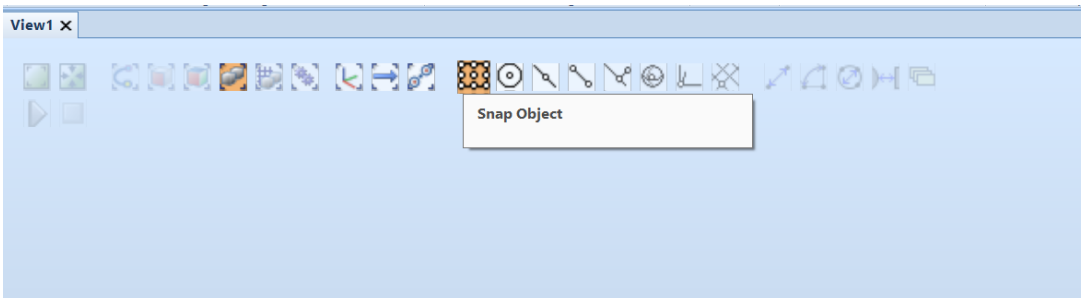


Рисунок 37. Меню инструментов

Устанавливают курсор в поле первой точки (первой колонки) (рис. 38).

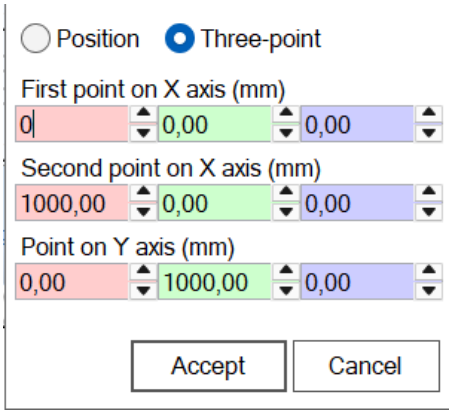


Рисунок 38. Вкладка «Frame by points» с установленным курсором в поле «X» координаты первой точки для задания системы координат рабочего объекта

Указывают первую точку в углу стола, вторую — впереди (по направлению на наблюдателя), третью — справа (рис. 39).

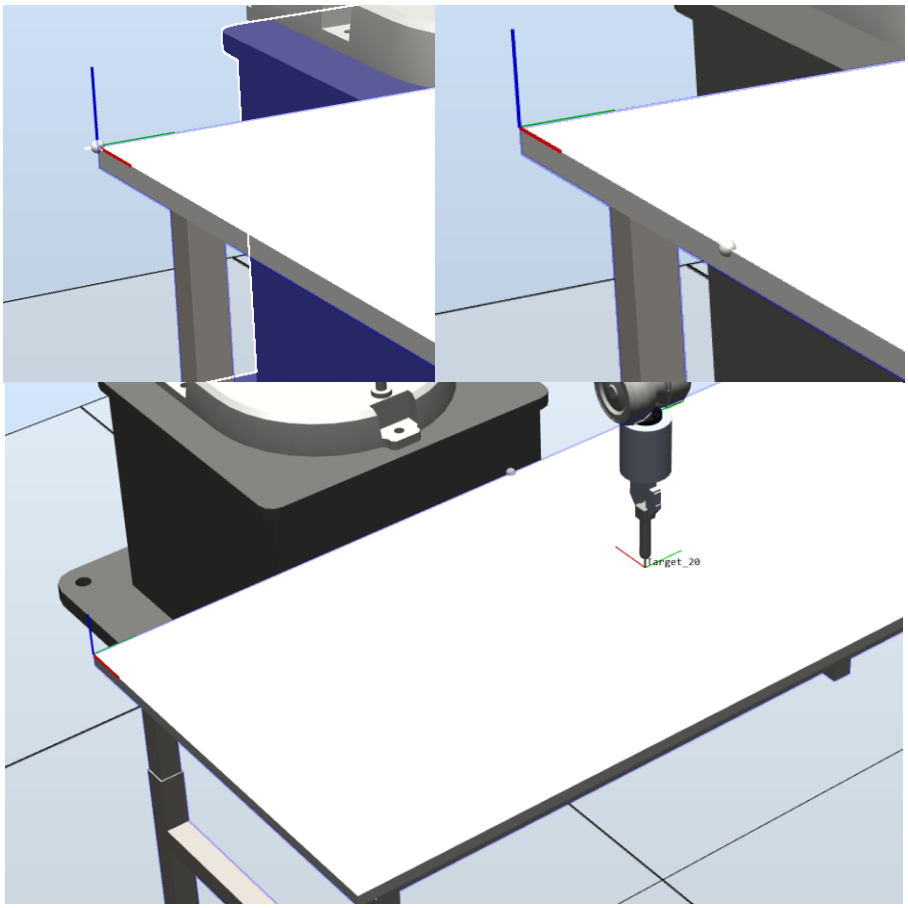


Рисунок 39. Точи в виртуальной модели ячейки, выбранные с помочью инструмента «Snap Object»

Выбор подтверждают нажатием Accept (рис. 40).

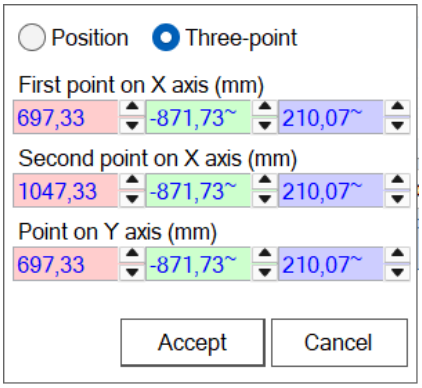


Рисунок 40. Вкладка «Frame by points» с координатами всех точек для формирования системы координат

При корректном выполнении формируется система координат, аналогичная базовой, но связанная с плоскостью стола (рис. 41).

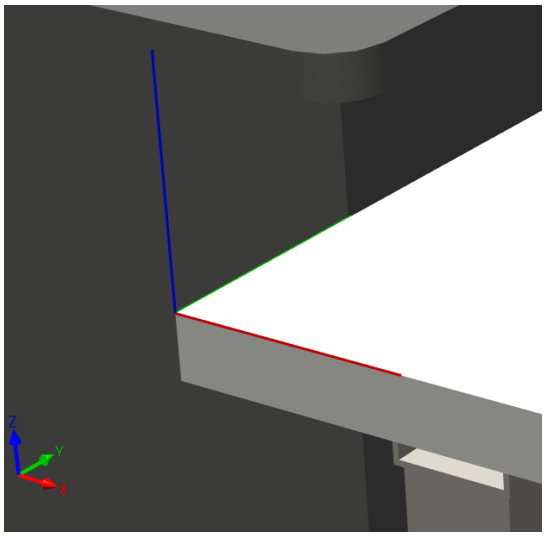


Рисунок 41. Результат создания системы координат рабочего объекта «Workobject»

Активный рабочий объект автоматически переключается на «wbj_Table» (рис. 42).

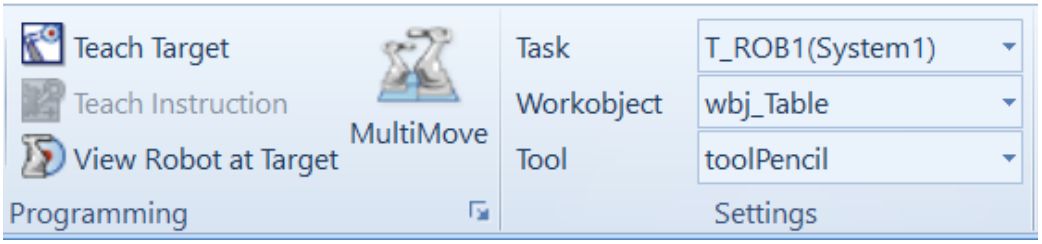


Рисунок 42. Меню активного рабочего объекта

Теперь все задаваемые точки определяются относительно стола. Во вкладке «wbj_Table» дерева проекта появляется первая заданная точка.

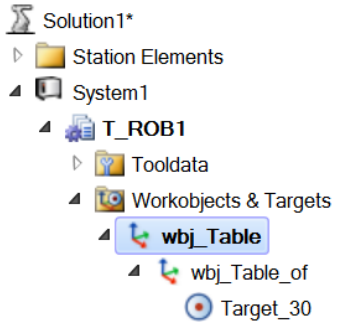


Рисунок 43. Дерево проекта

2.7. Задание целевых точек траектории

> *Траектория написания буквы складывается из точек, лежащих на плоскости стола (по ним выполняется вычерчивание), и точек подхода/отхода над столом. Совокупность таких точек однозначно определяет контур символа.*

Возвращаются к управлению роботом и перемещают его в требуемые положения для задания точек. Потянув мышкой за стрелки соответствующих цветов, перемещают инструмент манипулятора в нужно направлении (рис. 44).

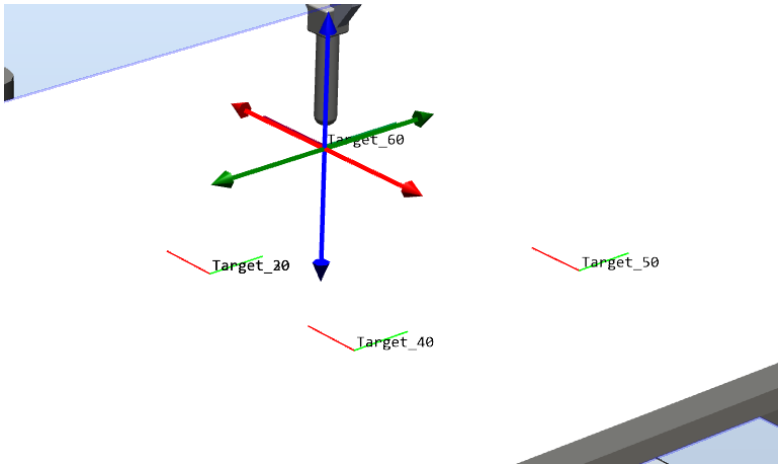


Рисунок 44. Инструмент манипулятора с активным инструментом линейного перемещения «Linear Jog»

Создают четыре точки, образующие контур буквы (рис. 45).

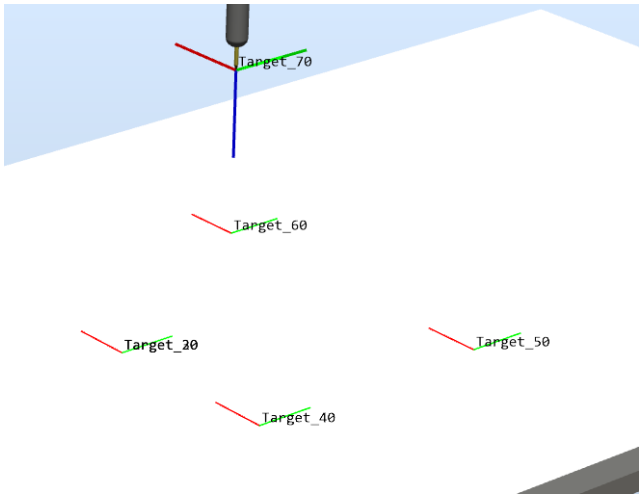


Рисунок 45. Записанные точки в плоскости стола

Пятую точку задают над столом — как точку выхода (отхода) из рабочего пространства по завершении вычерчивания.

2.8. Формирование путей движения и выбор типа интерполяции

> *Заданные целевые точки объединяются в пути (Path) — упорядоченные последовательности инструкций движения. На этом этапе для каждого сегмента выбирается тип интерполяции (MoveJ или MoveL) и параметры скорости и зоны.*

Точки добавляют в траекторию, чтобы робот мог последовательно их отработать. Для работы с путями переходят во вкладку «Path & Targets» (рис. 46).

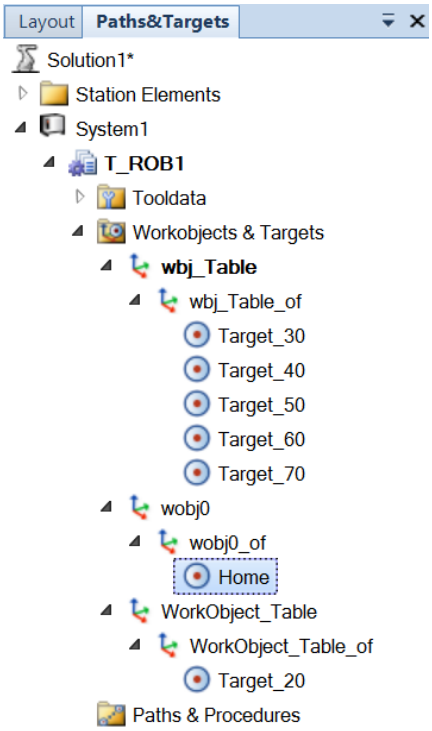


Рисунок 46. Вкладка «Path & Targets» со списком записанных точек

Раскрывают дерево до нужных точек, на точке «Home» вызывают контекстное меню (правая кнопка мыши) и выбирают «Add to new path».

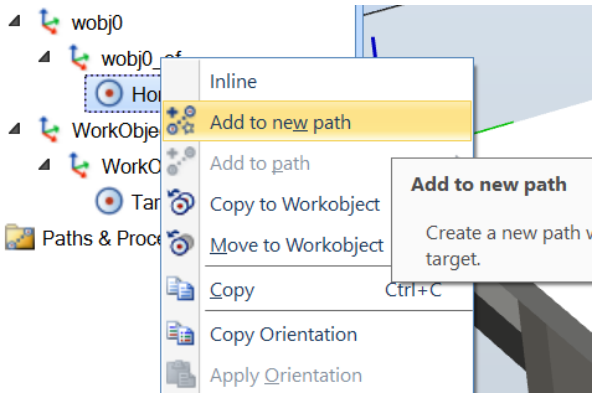


Рисунок 47. Дерево проекта с контекстным меню с выбранной инструкцией «Add to new path»

Автоматически генерируется путь «Path_10», содержащий инструкцию «MoveL Home» (рис. 48).

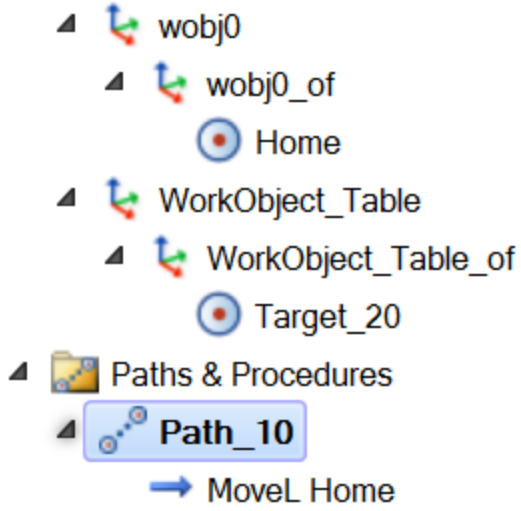


Рисунок 48. Дерево проекта с вкладкой «Path & Procedures»

При наведении на инструкцию отображаются её параметры, доступные для редактирования в нижней части окна (рис. 49).

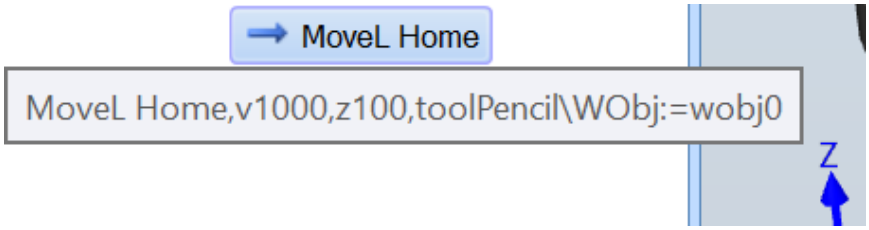


Рисунок 49. Всплывающее окно с параметрами инструкции «MoveL»

Для начала задают скорость «v100», а параметр «zone» заменяют на «fine» (точная остановка в точке). Можно использовать шаблон справа в нижней части окна приложения (рис. 50)

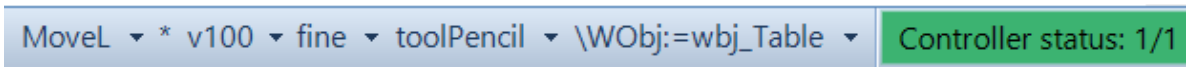


Рисунок 50. Шаблон с параметрам по умолчанию для инструкций движения

Траекторию переименовывают в осмысленное имя (например, go_Home), см. рис. 51.

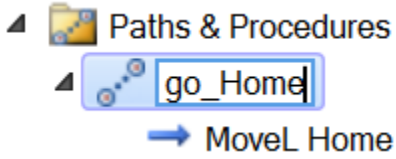


Рисунок 51. Редактирование имени процедуры в дереве проекта

Тип движения первой инструкции изменяют с «MoveL» на «MoveJ», чтобы исключить проход через сингулярные конфигурации при подходе из произвольного положения (рис. 52).

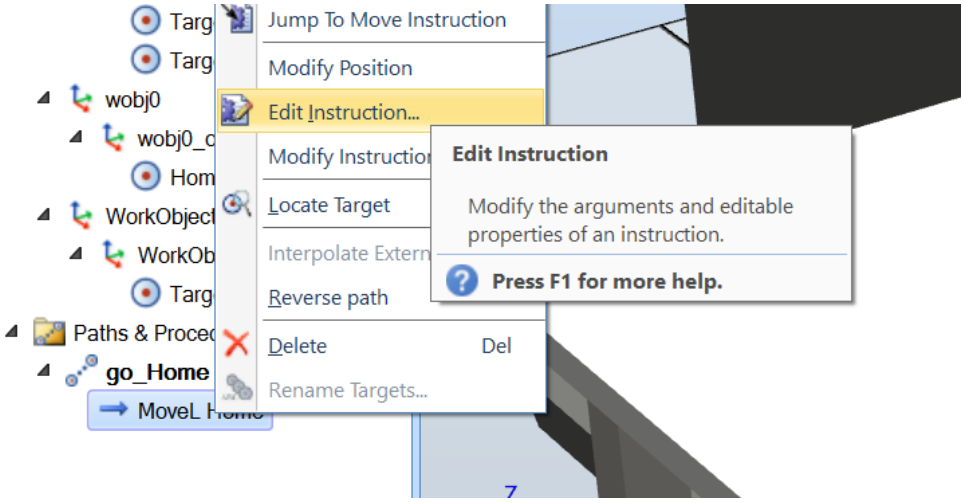


Рисунок 52. Дерево проекта с контекстным меню с выбранной инструкцией «Edit Instruction»

Открывается меню параметров инструкции движения, см. рис. 53. Для сохранения изменений, необходимо нажать кнопку «Apply».

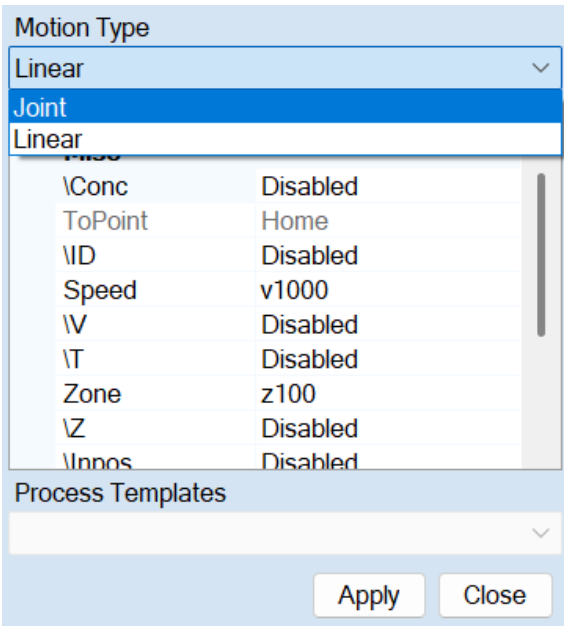


Рисунок 53. Меню параметров инструкции движения

Выделяют все пять точек, удерживая клавишу «Shift» и щёлкнув по первой и последней (рис. 54).

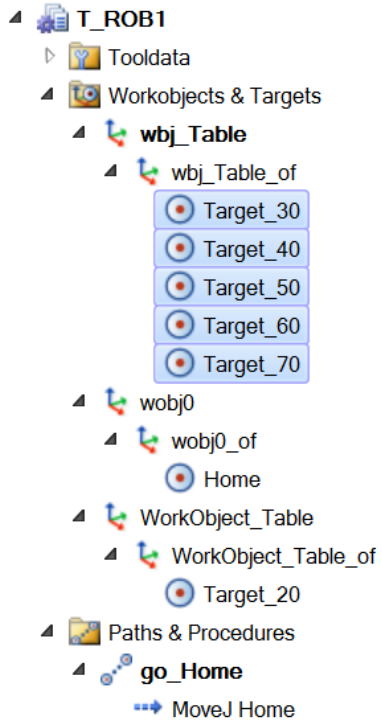


Рисунок 54. Дерево проекта с выбранными точками для генерации траектории

Выделенные точки добавляют в новый путь (например, Path_Letter), см. рис. 55.

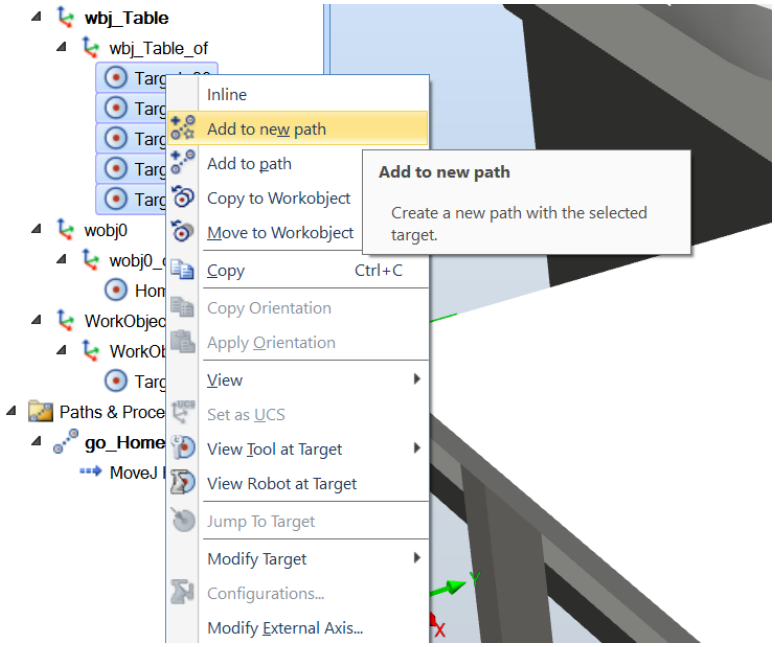


Рисунок 55. Дерево проекта с контекстным меню с выбранной инструкцией «Add to new path»

Тип движения **первой** инструкции этого пути также изменяют на «MoveJ».

2.9. Синхронизация с кодом RAPID и точка входа main

> *Графически построенная траектория преобразуется в исполняемый код RAPID операцией синхронизации (Synchronize to RAPID). Для запуска программы создаётся процедура-точка входа main, в которой пути вызываются в требуемом порядке.*

Сохраняют проект, чтобы зафиксировать текущее состояние станции. Выполняют команду «Synchronize to RAPID» для переноса объектов станции в код, см. рис. 56.

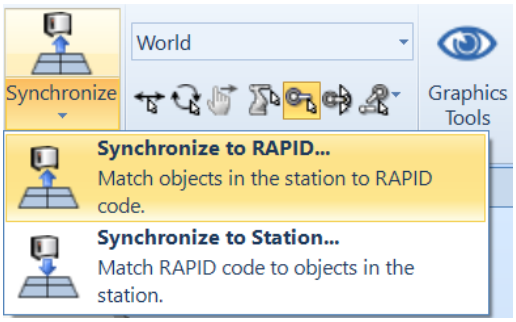


Рисунок 56. Меню синхронизации виртуальной модели ячейки с кодом на языке RAPID

Параметры синхронизации подтверждают нажатием «OK» (рис. 57).

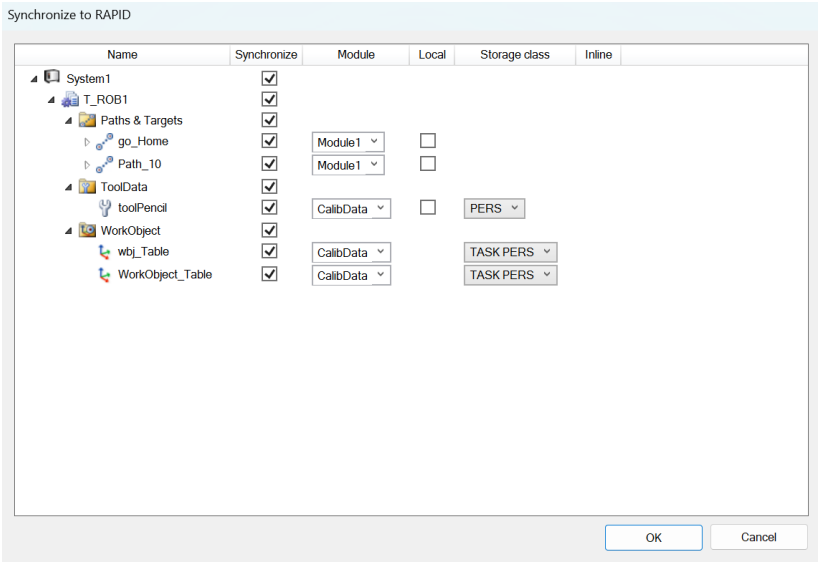


Рисунок 57. Меню параметров синхронизации с кодом RAPID

Во вкладке RAPID раскрывают структуру проекта и просматривают сгенерированные модули с объявлениями точек и процедурами путей, см. рис. 58.

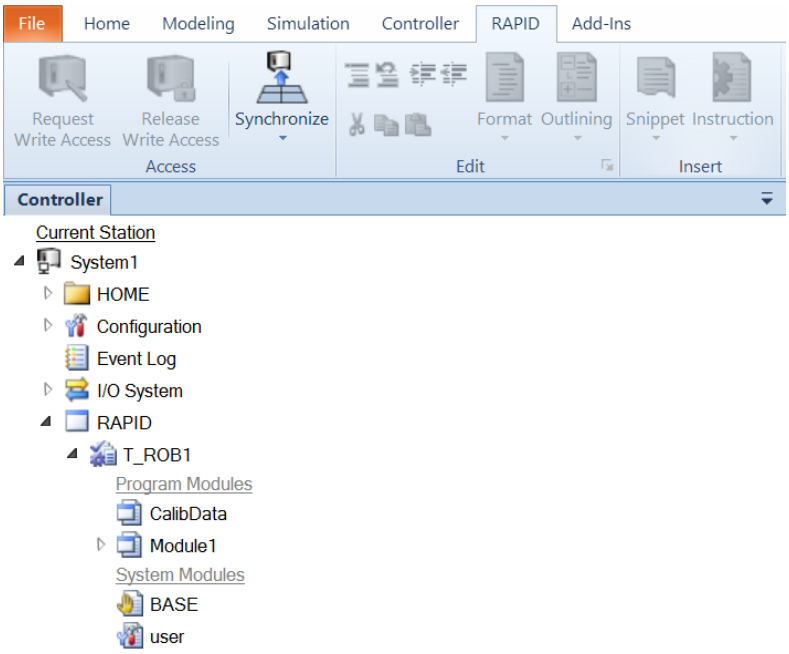


Рисунок 58. Структура проекта во вкладке «Controller» вкладки «RAPID»

Создают новый пустой путь, который станет основным циклом программы, см. рис. 59

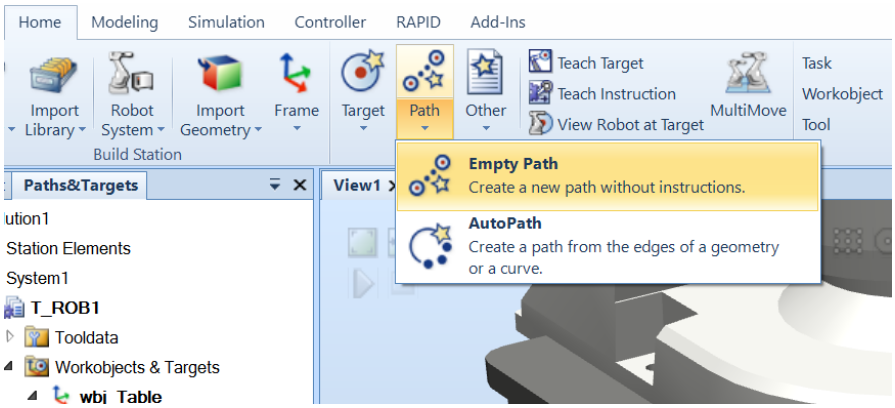


Рисунок 59. Кнопка «Empty Path»

Этот путь переименовывают на «main» — он является точкой входа в программу (рис. 60).

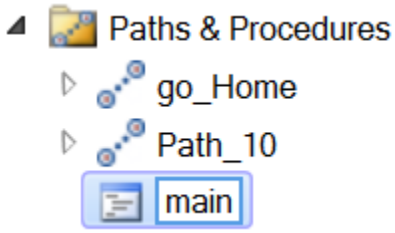


Рисунок 60. Дерево проекта с новой процедурой «main»

Левой кнопкой мыши перетаскивают созданные пути «go_Home», «Path_Letter» и повторно «go_Home» в процедуру «main», формируя цикл: подход в домашнюю точку — вычерчивание буквы — возврат в домашнюю точку, см. рис. 61.

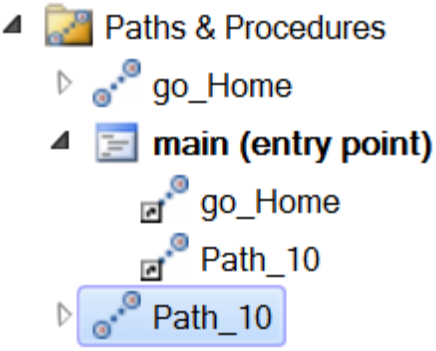


Рисунок 61. Дерево проекта с процедурой «main», содержащей пути «go_Home», «Path_Letter» и повторно «go_Home»

Повторно выполняют синхронизацию с RAPID, чтобы зафиксировать изменения в коде.

2.10. Запуск симуляции и визуализация траектории (TCP Trace)

> *Режим симуляции исполняет программу на виртуальном контроллере, что позволяет проверить корректность траектории до загрузки на реальное оборудование. Функция TCP Trace визуализирует путь, пройденный характеристической точкой инструмента.*

Для запуска симуляции переходят во вкладку «Simulation» и нажимают «Play», см. рис. 62.

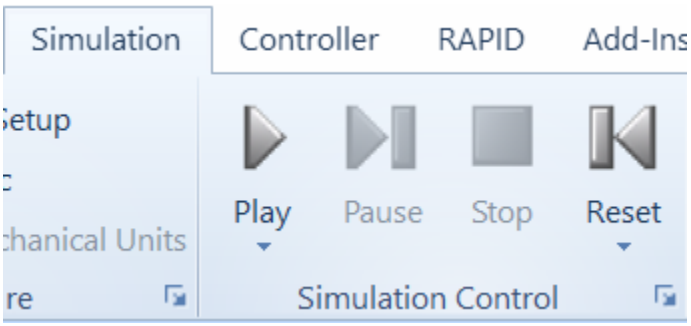


Рисунок 64. Вкладка «Simulation»

Для отслеживания траектории во вкладке «Simulation» выбирают инструмент «TCP Trace» (рис. 65).



Рисунок 65. Инструмент «TCP Trace» во вкладке «Simulation»

В появившемся слева окне устанавливают флаг «Enable TCP Trace», см. рис. 66. В поле «Primary color» задают цвет отображаемой траектории.

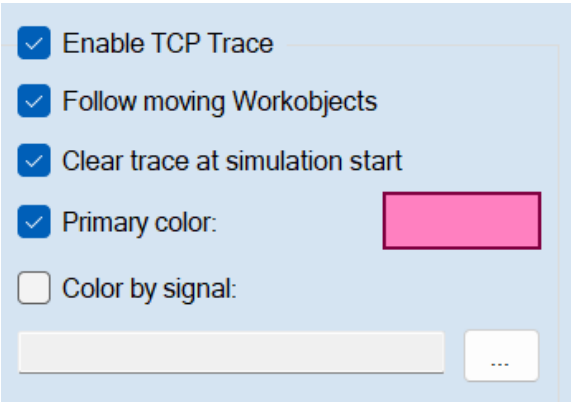


Рисунок 66. Меню инструмента «TCP Trace»

При запуске симуляции траектория движения TCP отображается выбранным цветом, что позволяет визуально оценить корректность вычерчивания, см. рис. 67.

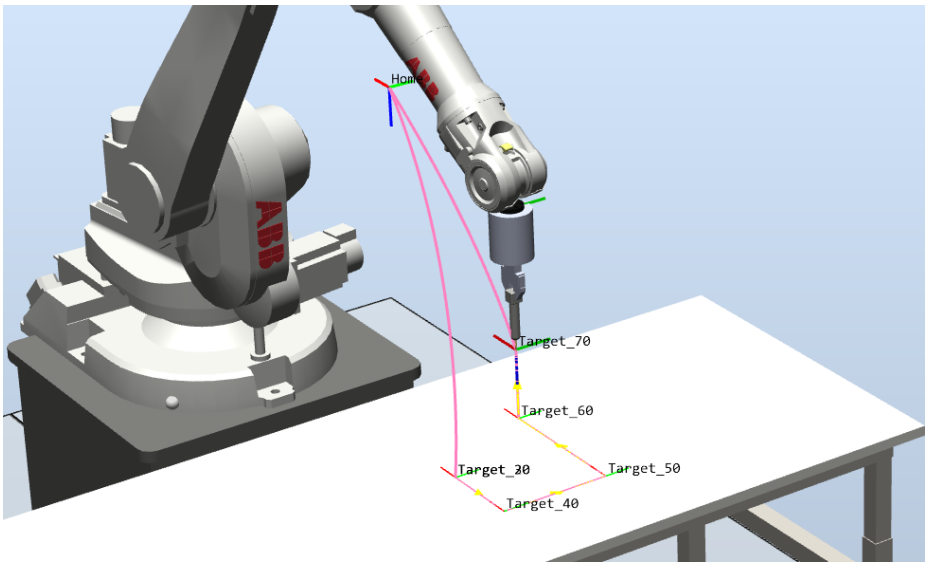


Рисунок 67. Результат работы инструмента «TCP Trace» при движении по запрограммированной траектории

Полезный приём: чтобы приподнять все точки для наглядного просмотра траектории, выполняют следующие действия. На рабочем объекте «wbj_Table» вызывают контекстное меню (правая кнопка мыши) и выбирают «Modify Workobject», см. рис. 68.

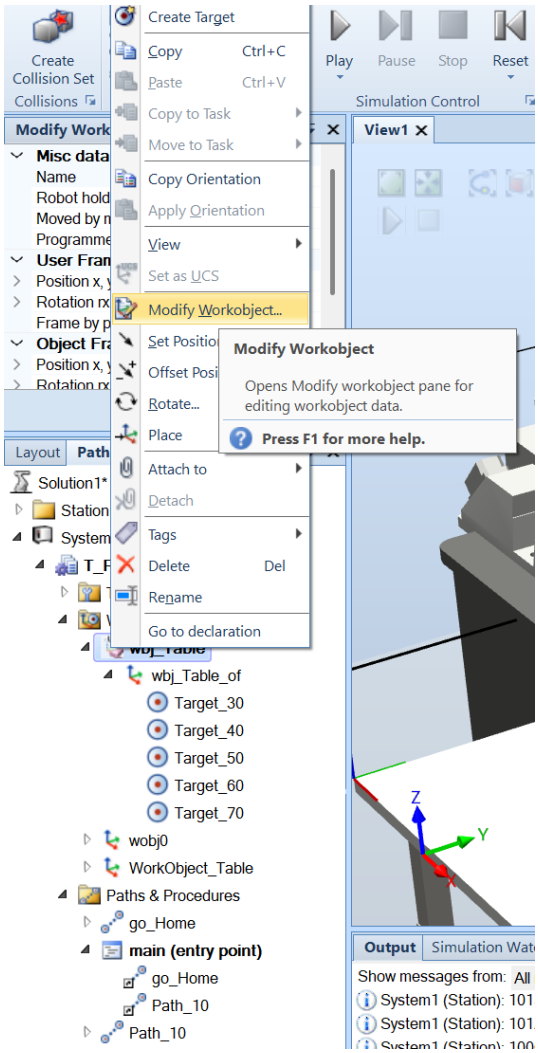


Рисунок 68. Дерево проекта с контекстным меню с выбранной инструкцией «Modify Workobject»

В открывшемся окне необходимо выбрать поле «Position x,y,z», см. рис. 69.

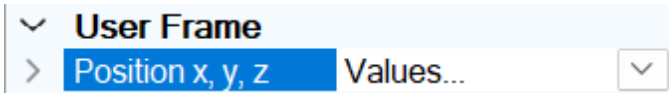


Рисунок 69. Окно редактирования системы координат рабочего объекта

Значение координаты по оси Z изменяют на 20–30 мм (рис. 70).

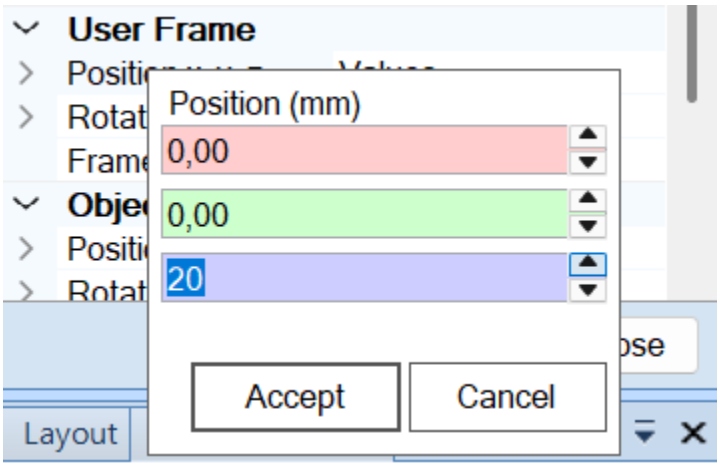


Рисунок 70. Окно для изменения позиции начала системы координат рабочего объекта

Все точки траектории смещаются вверх вместе с рабочим объектом (рис. 71). Однако следует учитывать: если сохранить проект и заново запустить симуляцию, робот будет исполнять ранее синхронизированную траекторию, а не текущую — для применения изменений требуется повторная синхронизация.

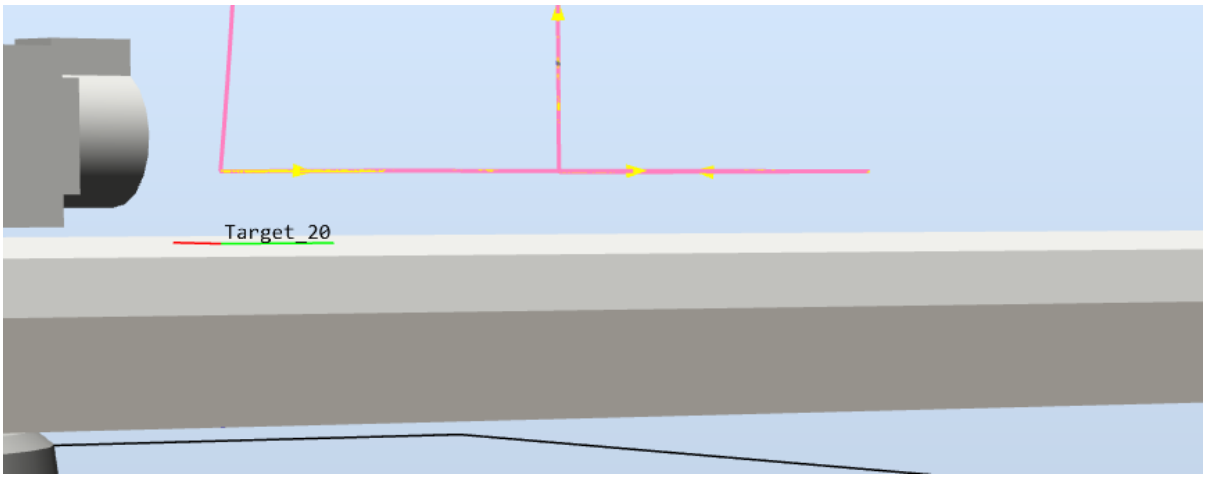


Рисунок 71. Новая траектория после изменения системы координат рабочего объекта

2.11. Сглаживание траектории

> *Параметр сглаживания (zone) позволяет роботу проходить промежуточные точки по дуге, не выполняя полной остановки. Это*

повышает плавность и скорость движения за счёт некоторого снижения геометрической точности в данной точке (см. п. 1.5).

Для создания скругления в выбранной точке выполняют следующие действия. Выбирают точку пути, для которой требуется скругление, и вызывают контекстное меню нажатием правой кнопки мыши, см. рис. 73.

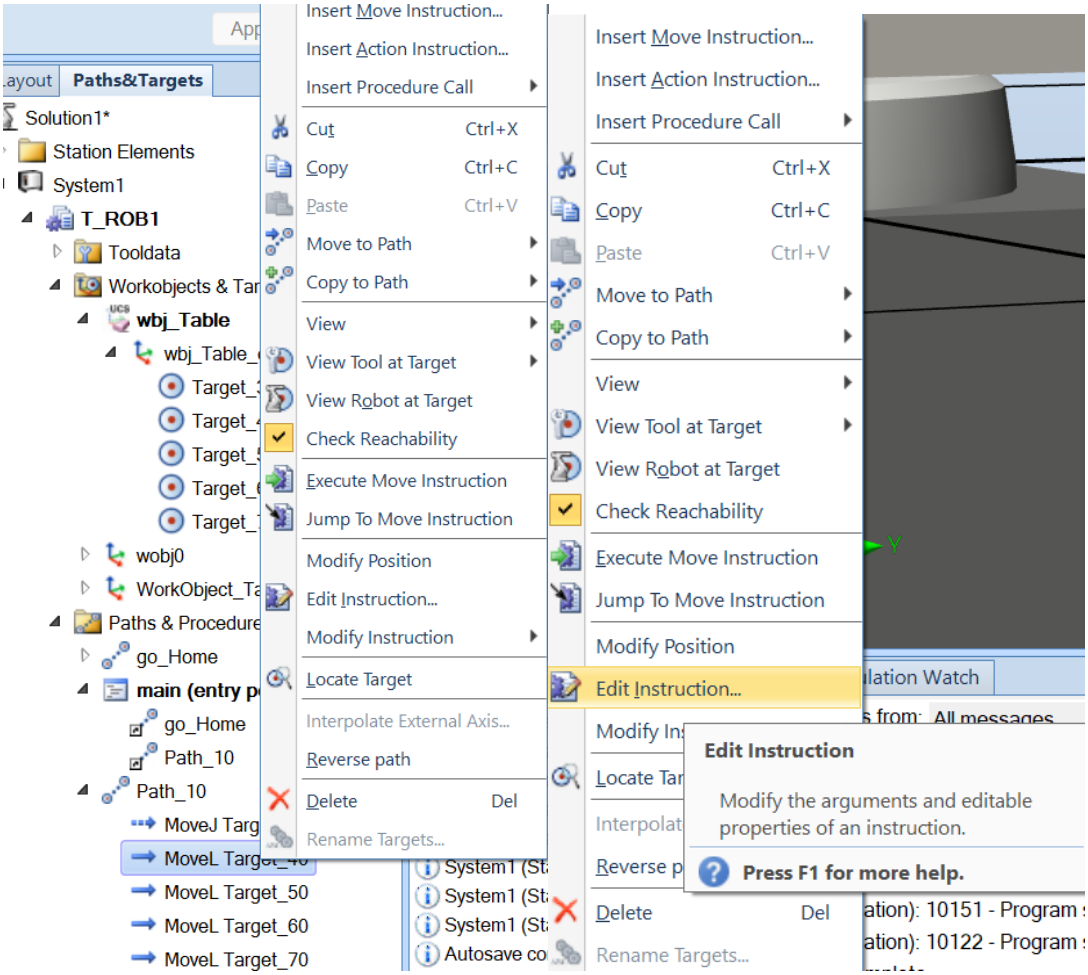


Рисунок 73. Дерево проекта с контекстным меню с выбранной инструкцией «Edit Instruction» для изменения параметра сглаживания траектории

Переходят в пункт Edit Instruction, см. рис. 74

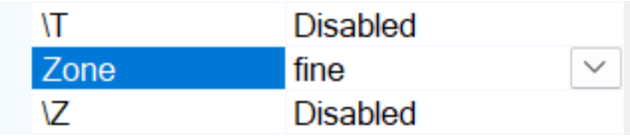


Рисунок 74. Поле «Zone» меню инструмента «Edit Instruction»

Изменяют параметр сглаживания (zone), например, на z100 — радиус зоны сглаживания 100 мм.

После этого обязательно сохраняют проект и выполняют синхронизацию с RAPID (рекомендуется установить все флаги синхронизации), чтобы изменения гарантированно попали в исполняемый код.

Повторно запускают симуляцию для проверки результата, см. рис. 75

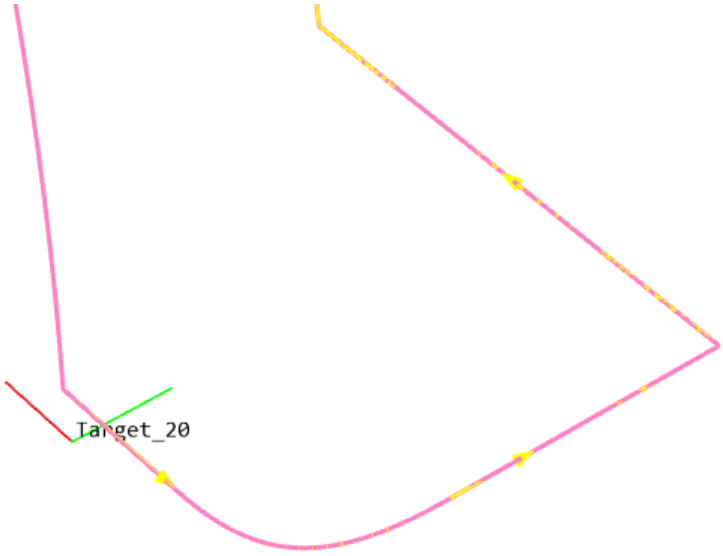


Рисунок 75. Результат изменения параметра сглаживания «zone»

В точке, где было задано скругление, изменение траектории становится отчётливо заметным.

2.12. Структура кода RAPID: robtarget, кватернионы и конфигурация

> *Анализ сгенерированного кода RAPID помогает понять, как графически заданная траектория представлена внутри программы. Ниже рассмотрены ключевые поля данных типа «robtarget» (см. п. 1.6).*

Числовые значения целевой точки в коде RAPID: выделенные зеленым значения соответствуют координатам положения TCP по осям X, Y, Z (в миллиметрах) и приведены в Листинге. 1.

Листинг 1. Объявление переменой типа «robtarget», зеленым цветом выделено поле положения инструмента.

```
CONST robtarget home:= [
[1082.239522999, 0, 849.419411487],
[0.069756473, 0, 0.99756405, 0],
[0, 0, 0, 0],
[9E+09, 9E+09, 9E+09, 9E+09, 9E+09,9 E+09]];
```

Следующая группа значений задаёт ориентацию инструмента в форме единичного кватерниона, см. Листинг 2

Листинг 2. Объявление переменой типа «robtarget», зеленым цветом выделено поле ориентации инструмента.

```
CONST robtarget home:= [
[1082.239522999, 0, 849.419411487],
[0.069756473, 0, 0.99756405, 0],
[0, 0, 0, 0],
[9E+09, 9E+09, 9E+09, 9E+09, 9E+09,9 E+09]];
```

Вектор конфигурации отражает способ решения обратной задачи кинематики — в каких конфигурациях достигается заданная точка, см. Листинг. 3

Листинг 3. Объявление переменой типа «robtarget», зеленым цветом выделено поле вектора конфигурации.

```
CONST robtarget home:= [
[1082.239522999, 0, 849.419411487],
[0.069756473, 0, 0.99756405, 0],
[0, 0, 0, 0],
[9E+09, 9E+09, 9E+09, 9E+09, 9E+09,9 E+09]];
```

Дополнительный вектор характеризует положение дополнительных осей, которыми может управлять манипулятор, например углами поворота станины (Литстинг 4).

Листинг 4. Объявление переменой типа «robtarget», зеленым цветом выделено поле.

```
CONST robtarget home:= [
[1082.239522999, 0, 849.419411487],
[0.069756473, 0, 0.99756405, 0],
[0, 0, 0, 0],
[9E+09, 9E+09, 9E+09, 9E+09, 9E+09,9 E+09]];
```

2.13. Сохранение программы для последующего переноса на робота

> *После успешной проверки в симуляции программа сохраняется на локальный компьютер или внешний носитель для последующей загрузки на контроллер реального робота. На этом этапе обязательна калибровка рабочего объекта с учётом фактического положения стола.*

Для переноса кода из RobotStudio на робота выполняют следующие действия. Во вкладке контроллера сохраняют программу (рис. 76).

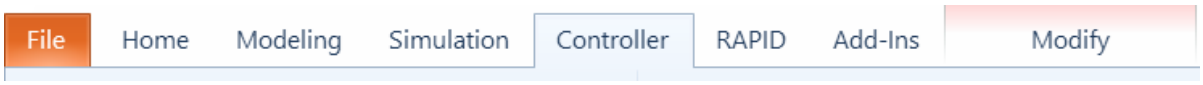


Рисунок 76. Вкладка «Controller»

В дереве проектов следует выбрать инструмент «Save program As..», см рис. 77.

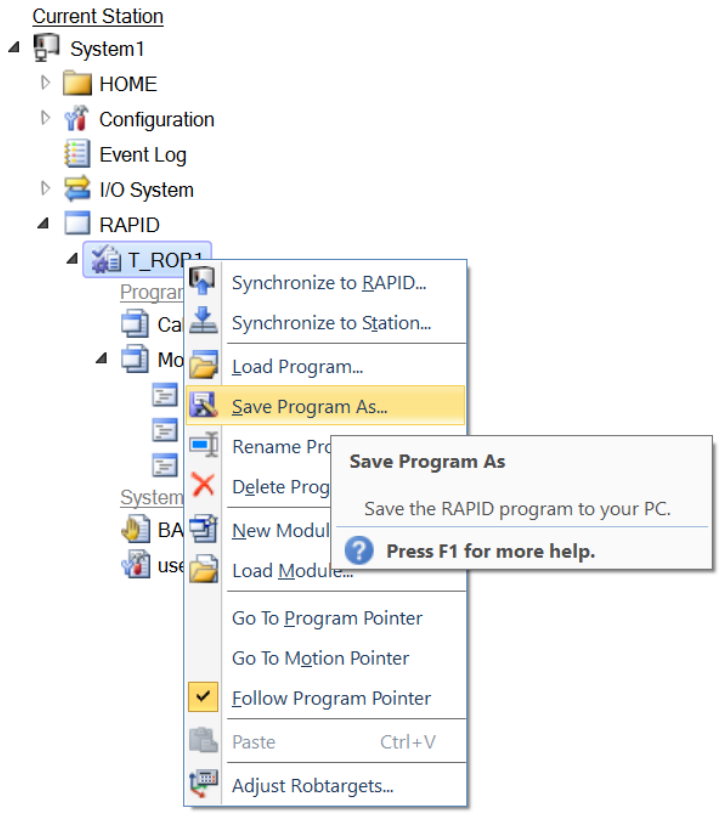


Рисунок 77. Дерево проекта с контекстным меню с выбранной инструкцией «Save Program As…»

Сохранять программу рекомендуется в отдельную папку «RAPID Programs» (рис. 78).

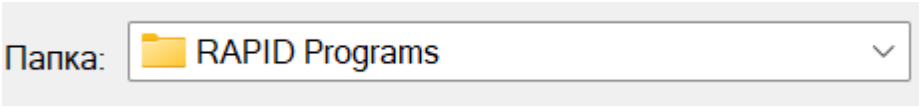


Рисунок 78. Рекомендованная директория для сохранения программ на языке RAPID

В указанном расположении формируется папка с файлами программы, которую переносят на внешний носитель для последующей загрузки на контроллер робота через пульт FlexPendant.

При первом запуске на реальном оборудовании необходимо проконтролировать положение TCP относительно рабочего объекта стола и

при необходимости скорректировать смещение системы координат по оси z (как правило, приподнять её на несколько десятков миллиметров, а затем точно опустить так, чтобы карандаш касался бумаги). Благодаря подпружинённому карандашу небольшие остаточные погрешности компенсируются автоматически.

3. Техника безопасности при работе с промышленным роботом

3.1. Общие положения

> *Промышленный робот-манипулятор ABB IRB 1600 является источником повышенной опасности. Его масса, скорость перемещения звеньев и усилие, развиваемое приводами, способны нанести серьёзные травмы человеку и привести к повреждению оборудования. В связи с этим к работе с роботом допускаются исключительно лица, прошедшие специальный инструктаж по технике безопасности и охране труда, а также ознакомленные с настоящим методическим пособием.*

Перед началом любых действий — будь то программирование в RobotStudio или отладка на реальном оборудовании — обучающийся **обязан**:

— убедиться в исправности робота и отсутствии видимых повреждений кабелей, разъёмов и механических частей;

— проверить работоспособность кнопки аварийной остановки (E-Stop);

— убедиться, что рабочая зона робота свободна от посторонних предметов и людей;

— надеть спецодежду, не имеющую свисающих элементов (галстуки, шарфы, длинные рукава), которые могут быть захвачены движущимися частями.

Категорически запрещается:

— работать с роботом в состоянии алкогольного или наркотического опьянения;

— оставлять робота без присмотра во включённом состоянии;

— вносить изменения в программу, не убедившись в безопасности последствий;

— находиться в рабочей зоне робота при работе в автоматическом режиме.

3.2. Аварийные ситуации

> *Кнопка аварийной остановки (E-Stop) рис. 79 расположена на пульте ручного управления FlexPendant, а также на лицевой панели шкафа контроллера. При нажатии E-Stop происходит немедленное отключение питания приводов и полная остановка всех движений робота. Это — основной и наиболее быстрый способ предотвращения аварии.*



Рисунок 79. Кнопка аварийной остановки (E-Stop) на лицевой панели шкафа контроллера и на пульте FlexPendant, выделены красным контуром

Порядок действий при аварийной ситуации:

— Немедленно нажать кнопку E-Stop (не отпуская до полной остановки робота).

— Убедиться, что робот полностью остановился и не совершает самопроизвольных движений.

— Оценить обстановку: наличие пострадавших, повреждений оборудования, возгорания.

— При необходимости вызвать помощь и сообщить руководителю.

— Не предпринимать попыток самостоятельного сброса ошибки до выяснения причин и разрешения преподавателя.

Сброс ошибки после E-Stop выполняется только после устранения причины аварии. Для этого необходимо:

— отпустить кнопку E-Stop (повернуть её по часовой стрелке);

— нажать кнопку «Motors» на контроллере или в интерфейсе RobotStudio;

— убедиться, что контроллер перешёл в состояние «Motors ON»;

— продолжить работу с пониженной скоростью, предварительно проверив траекторию.

3.3. Режимы работы и зоны безопасности

Робот ABB IRB 1600 может работать в двух основных режимах:

— «Ручной» режим — скорость движения ограничена 250 мм/с. Используется для отладки, калибровки и проверки траектории. В этом режиме оператор может находиться рядом с роботом и управлять им вручную или запускать программу пошагово через FlexPendant.

— «Auto» (автоматический) режим — робот работает по программе на полной скорости. Нахождение людей в рабочей зоне категорически запрещено.

Переключение между режимами осуществляется поворотом ключа на контроллере робота (рис.80). При этом необходимо убедиться, что все люди покинули рабочую зону.



Рисунок 80. Ключ на контроллере робота, выделен красным контуром

Рабочая зона робота IRB 1600-10/1.45 имеет радиус 1,45 м и ограничена сферой вокруг основания манипулятора. Для обеспечения безопасности рекомендуется:

— установить защитное ограждение по периметру рабочей зоны;

— оборудовать зону световой сигнализацией (жёлтый — «ручной» режим, зелёный — Auto, красный — авария);

— разместить кнопки E-Stop в доступных местах внутри и снаружи ограждения;

— нанести на пол разметку, обозначающую границу рабочей зоны.

3.4. Специфика работы с подпружиненным карандашом

> *Использование подпружиненного карандаша в качестве инструмента вносит дополнительные факторы риска. При неверной калибровке рабочего объекта по оси Z возможно:*

— слишком глубокое погружение карандаша, приводящее к поломке грифеля;

— повреждение поверхности стола;

— деформация пружины и потеря упругих свойств;

— неконтролируемое смещение инструмента.

Во избежание этих ситуаций необходимо:

— тщательно калибровать рабочего объекта с учётом фактической высоты стола;

— начинать отладку на пониженной скорости (v50–v100);

— контролировать степень сжатия пружины визуально;

— при первых признаках чрезмерного усилия немедленно остановить робота.

Запрещается:

— вручную поправлять траекторию или инструмент во время движения робота;

— касаться карандаша при работающих приводах;

— оставлять карандаш в контакте с бумагой при длительных паузах (возможна деформация пружины).

3.5. Действия после завершения работы

По окончании работы необходимо:

— Перевести робота в «ручной» режим.

— Установить манипулятор в домашнюю позицию (Home).

— Отключить питание приводов (Motors OFF).

— Сохранить программу на внешний носитель.

— Убрать рабочее место, удалить стружку и пыль.

— Сообщить преподавателю о завершении работы.

Соблюдение этих простых правил позволяет свести к минимуму риск травм и повреждений оборудования, а также обеспечивает долговременную и стабильную работу лабораторного комплекса.

4. Работа с пультом FlexPendant и перенос программы на реального робота

4.1. Интерфейс FlexPendant

> *FlexPendant — это промышленный пульт ручного управления роботом, работающий под управлением операционной системы RobotWare. Он представляет собой защищённый планшет с сенсорным экраном, джойстиком и набором физических кнопок (рис. 81).*

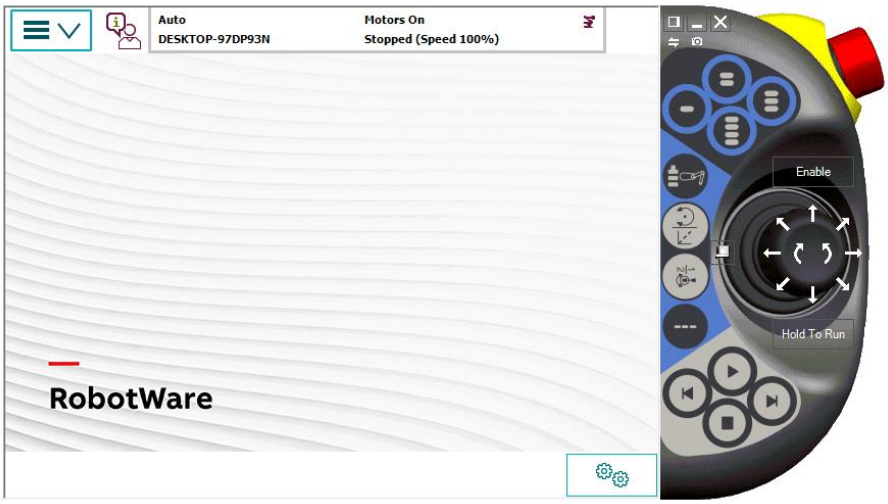


Рисунок 81. Пульт управления FlexPendant

Основные элементы FlexPendant:

— Джойстик — осуществляет ручное перемещение робота в «ручном» режиме. Позволяет управлять осями манипулятора (Jog Joint) или линейным перемещением TCP (Jog Linear).

— Кнопка разрешения движения — расположена на задней части пульта. Движение робота возможно только при её удержании. При отпускании робот немедленно останавливается. Имеет три положения (не нажато, средняя степень нажатия, максимальное нажатие). Первое и третье положение разрывают цепь подачи питания на двигатели.

— Кнопка E-Stop — красная кнопка на верхней панели для аварийной остановки.

— Сенсорный экран — отображает меню, программы, координаты, статус контроллера.

Навигация по меню (рис. 82) осуществляется касанием экрана. Основные разделы:

— ABB — главное меню, доступ к программам, калибровке, настройкам.

— Program Editor — редактор программ RAPID.

— Program Data — редактор переменных из программ RAPID.

— Jogging — ручное управление роботом.

— Calibration — калибровка осей робота.

— Event Log — журнал событий и ошибок.

— Inputs and Outputs — мониторинг и управление сигналами.

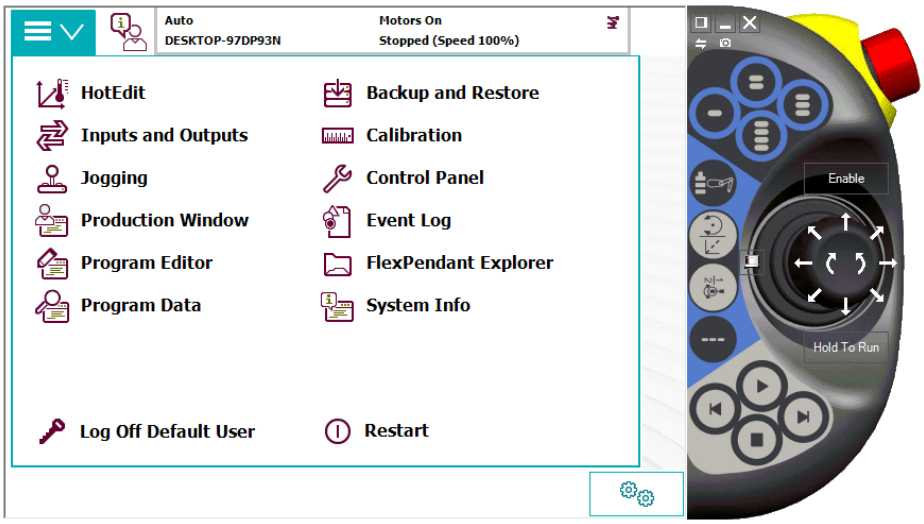


Рисунок 82. Навигация по меню пульта управления FlexPendant

4.2. Загрузка программы с USB-носителя

> *После успешной отладки программы в RobotStudio её необходимо перенести на реальный контроллер. Для этого используется USB-флешка, отформатированная в файловой системе FAT32.*

Порядок действий:

1. В RobotStudio открыть вкладку «Controller» → «RAPID» → «T_ROB1».

2. Вызвать контекстное меню (правая кнопка мыши) и выбрать «Save Program As»….

3. Указать папку «RAPID Programs» на локальном диске.

4. Скопировать файлы программы (модули .mod или архив .pgf) на USB-флешку. Рекомендуем скопировать папку с указанными файлами целиком.

5. Подключить флешку к разъёму USB на FlexPendant. Разъем находится справа в нижней части пульта.

6. На пульте перейти: «ABB» → «Program Editor» → «File» → «Load Module», см. рис. 83.

7. Выбрать файл модуля и подтвердить загрузку.

8. Проверить синтаксис: «Program Editor» → «Debug» → «Check Program», см. рис. 84.

9. Убедиться, что точка входа «main» определена корректно.

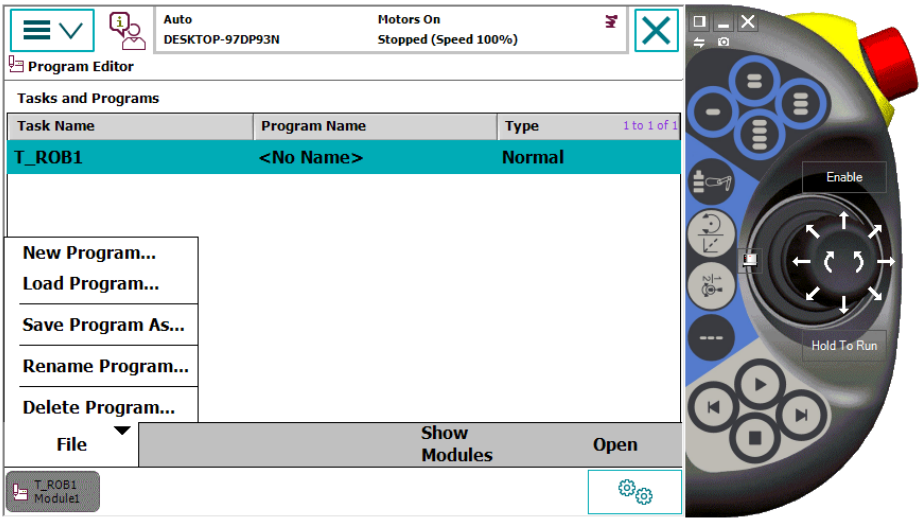


Рисунок 83. Загрузка программы с USB-носителя

Если при загрузке возникают ошибки, необходимо проверить:

— совместимость версии RobotWare;

— наличие всех объявленных данных (robtarget, tooldata, wobjdata);

— отсутствие дублирующихся имён модулей.

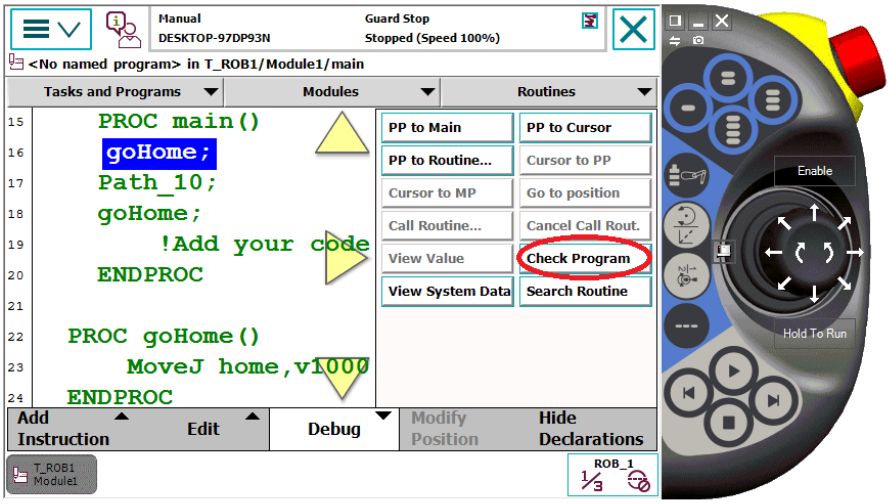


Рисунок 84. Проверка синтаксис программы на языке RAPID

4.3. Калибровка рабочего объекта (wobj) на реальном столе

> *Калибровка рабочего объекта — ключевой этап переноса программы на реальное оборудование. Она позволяет компенсировать расхождение между виртуальной моделью стола и его фактическим положением.*

Калибровка рабочего объекта — ключевой этап переноса программы на реальное оборудование. Она позволяет компенсировать расхождение между виртуальной моделью стола и его фактическим положением.

Порядок действий:

1. Убедиться, что робот находится в «ручном» режиме.

На FlexPendant перейти: «ABB» → «Program Data» → «wobjdata». Выбрать из списка типов переменных «wobjdata» и перейти к списку переменных двойным нажатием (рис. 85).

2. Выбрать нужную переменную, например «wobj_table» (или wbj_Table) и во вкладке «Edit» нажать Define, см. рис. 86.

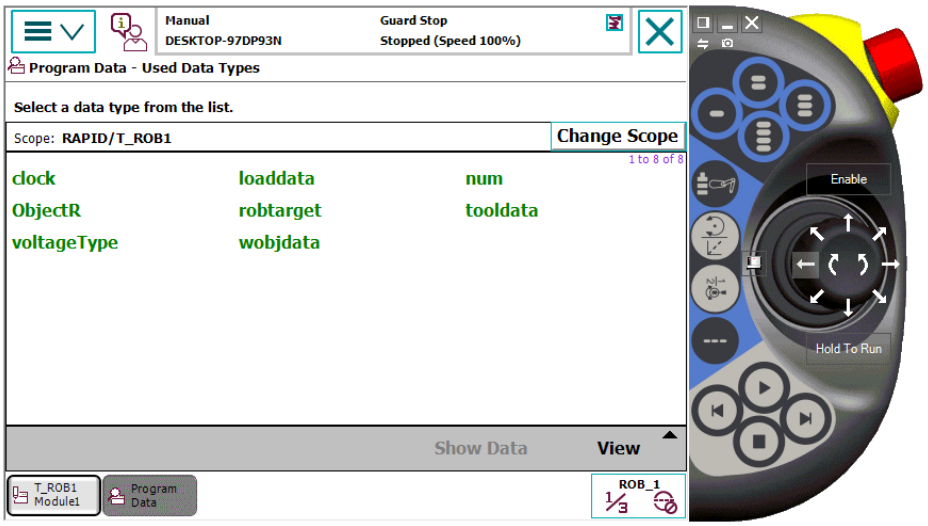


Рисунок 85. Раздел меню FlexPendant «Program Data»

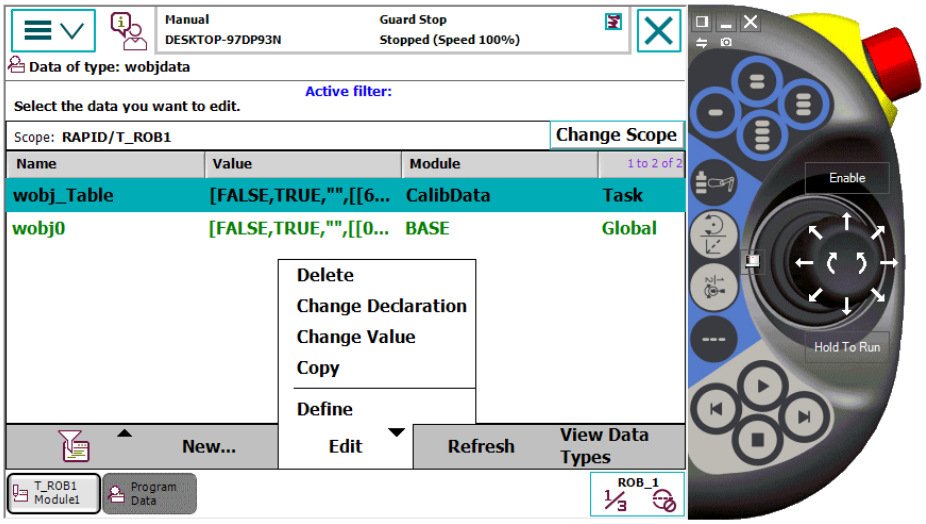


Рисунок 86. Редактирование системы координат рабочего объекта «Workobject»

3. Во вкладке «User method» из выпадающего списка выбрать метод «3 points» (по трём точкам), см. рис. 87

4. Подвести TCP карандаша к первой точке на столе (угол) и нажать «Modify Position».

5. Подвести ко второй точке (по направлению оси X) и зафиксировать, нажав «Modify Position»

6. Подвести к третьей точке (по направлению оси Y) и зафиксировать, снова нажав «Modify Position»

7. Нажать «OK» для сохранения.

8. Проверить смещение по оси Z во вкладке «Jogging» основного меню: при необходимости скорректировать значение вручную (обычно на 10–30 мм).

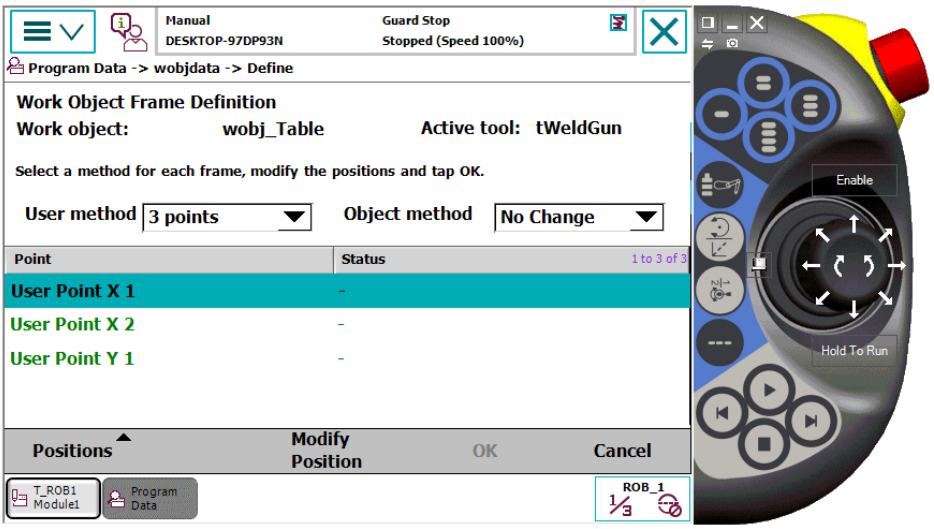


Рисунок 87. Редактирование системы координат рабочего объекта «User method»

После калибровки все точки траектории автоматически привязываются к фактическому положению стола. Это позволяет избежать пересчёта каждой точки при изменении высоты или наклона поверхности.

4.4. Калибровка инструмента (tooldata)

> *Для корректного позиционирования острия карандаша необходимо откалибровать инструмент, информация о котором сохранена в переменной toolPencil типа tooldata.*

Калибровка выполняется следующем способом:

1. Перейти: «ABB» → «Program Data» → «tooldata» → «Edit» →Define.

2. Выбрать метод TCP (default orient.), см. рис. 88.

3. Подвести острие карандаша к неподвижной реперной точке (например, вершине иглы) с четырёх разных ориентаций.

4. После каждой фиксации нажимать «Modify Position».

5. Нажать «OK» для сохранения.

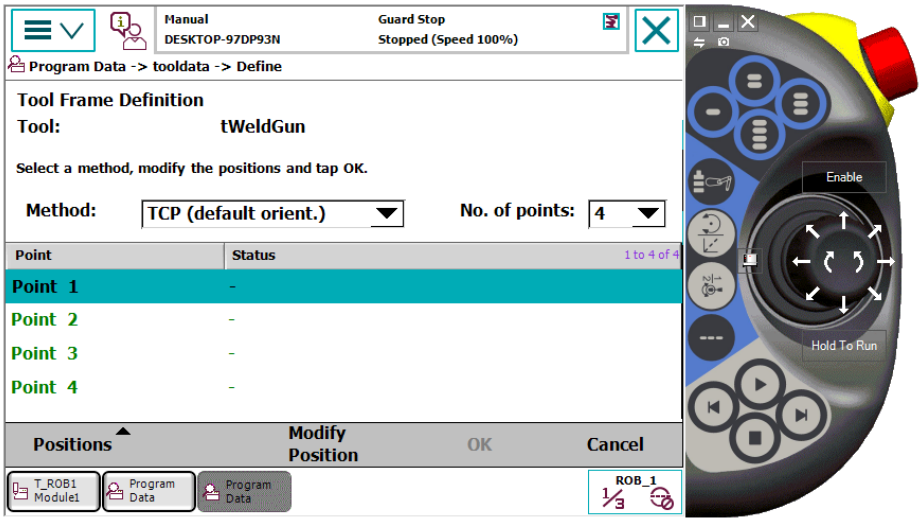


Рисунок 88. Калибровка инструмента (tooldata)

Проверка калибровки:

— подвести острие карандаша к реперной точке с разных ориентаций;

— убедиться, что отклонение не превышает 1–2 мм;

— при необходимости повторить калибровку.

4.5. Первый запуск программы на реальном роботе

> *После загрузки программы и калибровки можно приступить к первому запуску. Он должен выполняться в «ручном» режиме на пониженной скорости.*

Порядок действий:

1. Убедиться, что робот находится в домашней позиции.

2. На FlexPendant открыть программу во вкладке «Program Editor».

3. Во вкладке «Debug» нажать кнопку «PP to Main». Убедиться, что курсор в виде стрелки с левой стороны редактора указывает на первую инструкцию в процедуре «main» программы RAPID, см. рис. 89.

4. Нажать кнопку Play (или Start) для выполнения программы целиком. Либо кнопку «Стрелка вправо» для исполнения программы пошагово.

5. Удерживая кнопку «подачи питания», наблюдать за движением робота.

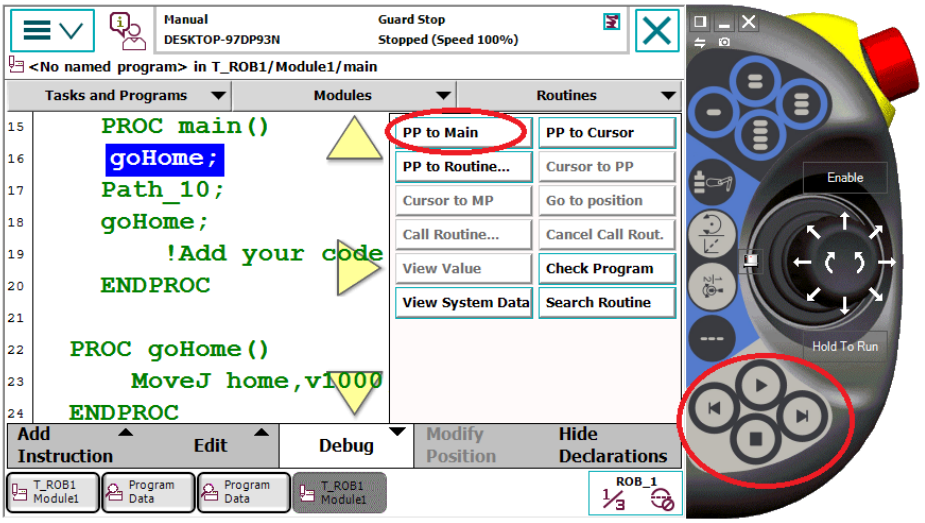


Рисунок 89. Запуск программы

При отклонении траектории немедленно отпустить кнопку — робот остановится.

Проверить касание карандаша к бумаге: при необходимости скорректировать смещение по Z.

Убедиться, что все буквы вычерчены корректно.

После успешной отладки в ручном режиме, можно перейти в режим Auto:

1. Убедиться, что все покинули рабочую зону.

2. Переключить ключ Mode в положение Auto.

3. Нажать на кнопку «Motors ON», см. рис. 90.

4. Запустить программу с пульта или внешнего сигнала.



Рисунок 90. Кнопка «Motor» на контроллере робота.

4.6. Возможные ошибки при переносе

> *При переносе программы на реального робота возможны следующие ошибки:*

При переносе программы на реального робота возможны следующие ошибки:

— «Out of reach» — точка недостижима. Решение: проверить координаты точки, изменить конфигурацию или положение рабочего объекта.

— Сингулярность — робот не может пройти через точку из-за совпадения осей. Решение: заменить «MoveL» на «MoveJ» для подхода, изменить конфигурацию (confdata).

— Коллизия — столкновение с препятствием. Решение: проверить «Collision Set», скорректировать траекторию.

— Программа не запускается — проверить точку входа main, права доступа, статус контроллера.

— Карандаш не касается бумаги — скорректировать смещение рабочей области «wobjdata» по Z.

— Карандаш пробивает бумагу — уменьшить глубину погружения, проверить калибровку TCP.

5. Возможности языка RAPID

> *Приложение на RAPID называют задачей* (TASK). *Задача состоит из одного и/или нескольких модулей* (MODULE). *Модуль, в свою очередь, содержит наборы данных и подпрограмм* (ROUTINE). *Модули могут быть программными (частью задачи) и системными. Модуль задачи считается частью задачи/приложения, а системный модуль — частью системы.*

Существует три типа подпрограмм: **процедуры** (procedures), **функции** (functions) и **подпрограммы-прерывания** (trap routines).

Существует четыре способа представления/хранения данных — **константы** (constants), **переменные** (variables), **постоянные данные** (persistents) и параметры. Отличие «persistents» от «variables»: переменная типа «persistents» запоминает в памяти свое последнее значение после выполнения программы.

Язык не чувствителен к регистру – буквы верхнего и нижнего регистра считаются одинаковыми.

Первым символом идентификатора должна быть буква. Другими символами могут быть буквы, цифры или символы подчеркивания «_». Максимальная длина любого идентификатора составляет 32 символа, каждый из которых является значащим.

Зарезервированные **ключевые** слова:

```
ALIAS, AND, BACKWARD, CASE, CONNECT, CONST, DEFAULT, DIV,
DO, ELSE, ELSEIF, ENDFOR, ENDFUNC, ENDIF, ENDMODULE,
ENDPROC, ENDRECORD, ENDTEST, ENDTRAP, ENDWHILE, ERROR,
EXIT, FALSE, FOR, FROM, FUNC, GOTO, IF, INOUT, LOCAL,
MOD, MODULE, NOSTEPIN, NOT, NOVIEW, OR, PERS, PROC,
RAISE, READONLY, RECORD, RETRY, RETURN, STEP, SYSMODULE,
TEST, THEN, TO, TRAP, TRUE, TRYNEXT, UNDO, VAR, VIEWONLY,
WHILE, WITH, XOR.
```

Язык программирования RAPID — это язык свободного формата, что означает, что пробелы можно использовать где угодно, за исключением: идентификаторов, зарезервированных слов, числовых значений, «placeholder».

Символы новой строки, табуляции и перевода страницы можно использовать везде, где допускается использование пробела, за исключением комментариев. Идентификаторы, зарезервированные слова и числовые значения должны быть отделены друг от друга пробелом, символом новой строки, символом табуляции или перевода страницы.

Комментарий начинается с восклицательного знака «!» и заканчивается символом новой строки. Он занимает оставшуюся часть строки (начиная с восклицательного знака) и не может встречаться вне объявления модуля (Листинг 5).

Листинг 5. Объявление комментария в коде программы RAPID.

```
! comment
```

«Placeholders» могут использоваться для временного представления частей программы, которые еще не определены. Программа, содержащая placeholder, является синтаксически корректной и может быть загружена в память программы.

<TDN> определение типа данных

<DDN> объявление данных

<RDN> объявление программы

<PAR> формальный необязательный альтернативный параметр

<ALT> необязательный формальный параметр

<DIM> формальная (соответствующая) размерность массива

<SMT> инструкция

<VAR> ссылка на объект данных (переменную, постоянную память или параметр)

<EIT> инструкции оператора условия «IF»

<CSE> инструкции оператора условия «TEST CASE»

<EXP> выражение.

<ARG> аргумент вызова процедуры

<ID> индетификатор

Программа разделена на программные модули и системные модули. На рис. 91 представлена структура программы языка RAPID.

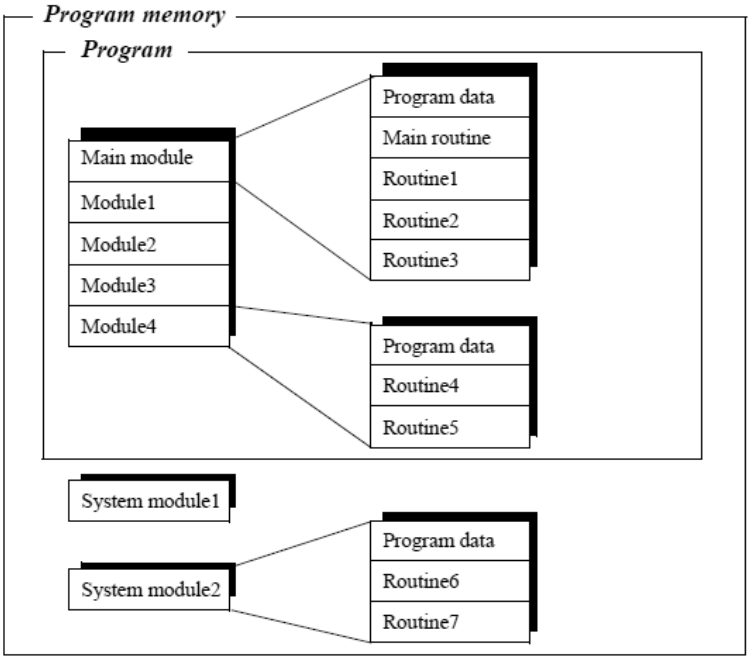


Рисунок 91. Структура программы языка RAPID

Программный модуль может состоять из различных данных и процедур. Каждый модуль или всю программу можно скопировать на диск и т.д., и наоборот. Один из модулей содержит процедуру входа, глобальную процедуру под названием «main». Выполнение программы означает, по сути, выполнение процедуры «main». Программа может включать множество модулей, но только один из них будет иметь основную процедуру.

Модуль может, например, определять интерфейс с внешним оборудованием или содержать геометрические данные, которые либо генерируются из систем САПР, либо создаются в режиме онлайн путем оцифровки (обучение программированию). В то время как небольшие установки часто содержатся в одном модуле, более крупные установки могут иметь основной модуль, который ссылается на процедуры и/или данные, содержащиеся в одном или нескольких других модулях.

Системные модули используются для определения общих, специфичных для системы данных и процедур, таких как инструменты «tools». Они не включаются при сохранении программы, а это означает, что любое обновление системного модуля повлияет на все существующие программы, находящиеся в данный момент или загруженные на более позднем этапе в память программы.

5.1. Типы данных переменных и констант

Выделяют три типа данных переменных и констант:

Базовый «atomic» тип является атомарным в том смысле, что он не определен на основе какого-либо другого типа и не может быть разделен на части или компоненты, например num, bool, string

Структура «record» — это составной тип с именованными упорядоченными компонентами, например pos. Компонент может иметь базовый тип или структуру. Значение данного типа может быть выражено с использованием совокупного представления, например [300, 500, depth] pos совокупное значение записи. Доступ к определенному компоненту данных записи можно получить, используя имя этого компонента, например pos1.x:= 300; Это означает присвоить x-компоненте переменной pos1 значение 300. Пример объявления структуры «object» приведен в Листинге 6.

Листинг 6. Объявление структуры «object».

```
RECORD object
num usecount;
string name;
```

```
ENDRECORD
VAR object obj1;
obj1.name := “Screw”
```

Псевдоним «alias» по определению равен другому типу. Типы псевдонимов позволяют классифицировать объекты данных. Пример объявления псевдонима «newTypeNum» приведен в Листинге 7.

Листинг 7. Объявление псевдонима «newTypeNum».

```
ALIAS <data type> <identifier>;
ALIAS num newTypeNum;
VAR num level;
VAR dionum high:=1;
level:= high;
```

Данная операция справедлива, так как «dionum» является типом данных - псевдоним «alias» для базового «atomic» типа данных «num».

Область видимости данных обозначает область, в которой данные видны. Необязательная директива «LOCAL» объявления данных классифицирует данные как локальные (внутри модуля), в противном случае они являются глобальными. Обратите внимание, что директиву «LOCAL» можно использовать только на уровне модуля, а не внутри процедуры. Пример объявления локальной переменной «local_variable» и глобальной переменной «global_variable» приведен в Листинге 8.

Листинг 8. Объявление локальной переменной «local_variable».

```
LOCAL VAR num local_variable;
VAR num global_variable;
```

Данные, объявленные вне подпрограммы, называются данными программы. К данным программы применяются следующие правила области видимости:

• Видимость предопределенных или глобальных данных программы может распространяться на любой модуль.

• Видимость данных локальной программы распространяется только на модуль, в котором они содержатся.

• В пределах своей области видимости данные локальной программы скрывают любые глобальные данные или подпрограммы с тем же именем, включая инструкции и предопределенные подпрограммы и данные (рис. 92).

Данные программы могут не иметь того же имени, что и другие данные или процедура в том же модуле. Глобальные данные программы могут не иметь того же имени, что и другие глобальные данные или процедура в другом модуле.

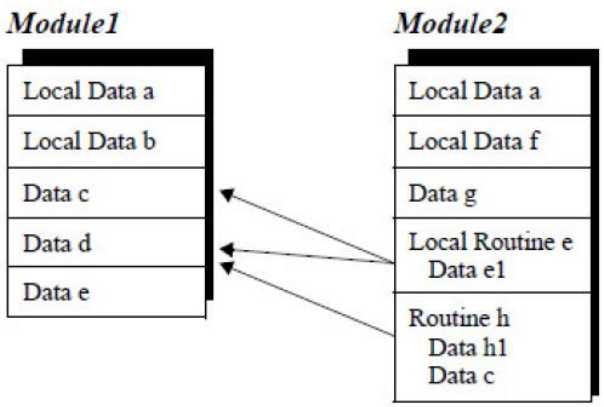


Рисунок 92. Области видимости данных разных модулей программы языка RAPID

5.2. Операторы и инструкции

При работе с переменными в RAPID используют следующие выражения:

Арифметические выражения:

`+, -, *, /, DIV, MOD.`

Логические выражения:

<, <=, =,>=, >, <>, AND, XOR, OR, NOT.

Сложение строк «string»:

«IN» + «PUT» = «INPUT»

Математические инструкции:

Incr: Инструкция для увеличения аргумента на 1

Decr: Инструкция для уменьшения аргумента на 1

Rand: Инструкция для возврата случайного числа

Abs: Инструкция для возврата модуля числа

Round: Инструкция для округления числа

Max: Инструкция для возврата максимального числа из двух

Min: Инструкция для возврата минимального числа из двух

Sqrt: Инструкция для возврата квадратного корня аргумента

Exp: Инструкция для возврата экспоненты аргумента

Pow: Инструкция для возврата аргумента в заданной степени

ACos, ASin: Инструкция для возврата arc cos и arc sin аргумента

Cos, Sin: Инструкция для возврата cos и sin аргумента

Условный оператор:

Начинается с «IF», заканчивается «ENDIF». Пример использования условного оператора приведен в Листинге 9.

Листинг 9. Использование условного оператора.

```
IF counter > 100 THEN
counter := 100;
ELSEIF counter < 0 THEN
counter := 0;
```

```
ELSE
counter := counter + 1;
ENDIF
```

Оператор цикла «FOR».

Начинается с «FOR», заканчивается «ENDFOR». Пример использования оператора цикла приведен в Листинге 10.

Листинг 10. Использование оператора цикла «FOR».

```
FOR i FROM 10 TO 1 STEP -1 DO
sum:= sum + i;
ENDFOR
```

Оператор цикла «WHILE».

Начинается с «WHILE», заканчивается «ENDWHILE». Пример использования оператора цикла приведен в Листинге 11.

Листинг 11. Использование оператора цикла «WHILE».

```
WHILE index < 100 DO
index := index + 1;
ENDWHILE
```

Условный оператор ветвления «TEST».

Начинается с «TEST», заканчивается «ENDTEST». Пример использования условного оператора ветвления «TEST» приведен в Листинге 12.

Листинг 12. Использование условного оператора ветвления «TEST».

```
TEST product_id
CASE 1:
```

```
proc1 x, y, z;
CASE 2: ...
proc2 x, y, z;
DEFAULT :
EXIT;
ENDTEST
```

Программа выполняется, как правило, последовательно, то есть инструкция за инструкцией. Иногда для обработки различных ситуаций, которые могут возникнуть во время выполнения, требуются инструкции, которые прерывают это последовательное выполнение и вызывают другую инструкцию.

Ход программы можно контролировать в соответствии с пятью различными принципами:

- Путем вызова другой подпрограммы (процедуры) и, когда эта подпрограмма была выполнена, продолжить выполнение с помощью инструкции, следующей за вызовом подпрограммы.

- Выполняя различные инструкции в зависимости от того, удовлетворено или нет заданное условие.

- Повторяя последовательность инструкций несколько раз или до тех пор, пока не будет выполнено заданное условие.

- Перейдя на конкретное место в рамках той же процедуры.

- Остановив выполнение программы.

Инструкции для движения робота:

`MoveC`: движение инструмента робота по кругу.

`MoveJ`: произвольное движение инструмента робота.

`MoveL`: движение инструмента робота по прямой линии.

`MoveAbsJ`: движение определенных сочленений робота.

Инструкции для общение через устройства ввода-вывода и файлы:

`TPErase`: инструкция для очистите дисплей оператора FlexPendant.

`TPWrite`: инструкция для выводы текста на дисплей оператора FlexPendant.

`TPReadNum`: инструкция для чтения числа от оператора FlexPendant.

`Open`: инструкция для открытия файла для чтения/записи.

`Write`: инструкция для записи в файл.

`Close`: инструкция для закрытия файла.

`ReadNum`: инструкция для чтения из файл числовых данных.

`ReadStr`: инструкция для чтения из файл строковых данных.

Задержки и синхронизация:

`WaitTime`: пауза на заданное время (в секундах).

`WaitDI`: ожидание дискретного сигнала.

`SetDO`: установка дискретного выхода.

`WaitUntil`: ожидание выполнения условия.

Эти инструкции полезны при интеграции робота с внешним оборудованием (конвейер, захват, система технического зрения).

5.3. Массивы

> *Переменная* «`VAR`»*, переменная* «`PERS`» *или константа* «`CONST`»*, объявленная как массив, может ссылаться на весь массив или на отдельный элемент.*

Пример объявления массива «array» приведен в Листинге 13.

Листинг 13. Объявление массива «array».

```
VAR num array{3};
```

Пример объявления массива точек приведен в Листинге 14.

Листинг 14. Объявление массива точек «pLetter».

```
CONST robtarget pLetter{5}:= [
[[100, 0], [1, 0, 0], [0, 0, 0], [0, 0, 0, 0, 0, 0]],
[[200, 0], [1, 0, 0], [0, 0, 0], [0, 0, 0, 0, 0, 0]],
];
```

Пример обхода массива в цикле приведен в Листинге 15.

Листинг 15. обхода массива в цикле.

```
FOR i FROM 1 TO 5 DO
    MoveL pLetter{i}, v100, fine, toolPencil;
ENDFOR
```

Массивы позволяют компактно описывать повторяющиеся траектории и легко изменять их параметры.

5.4. Обработчики прерывания и ошибок

> *Прерывания — это определяемые программой события, обозначаемые номерами прерываний. Прерывание происходит, когда условие прерывания истинно. В отличие от ошибок возникновение прерывания не связано напрямую (синхронно) с конкретной позицией кода.*

Возникновение прерывания приводит к приостановке нормального выполнения программы, и управление передается процедуре прерывания.

Несмотря на то, что робот немедленно распознает возникновение прерывания (с задержкой только из-за скорости аппаратного обеспечения), его ответ – вызов соответствующей процедуры прерывания – может иметь место только в определенных позициях программы, а именно:

— когда вводится следующая инструкция,

— в любой момент выполнения ожидающей инструкции, например «WaitUntil»,

— в любой момент выполнения инструкции перемещения, например «MoveL».

Обычно это приводит к задержке в 2–30 мс между распознаванием прерывания и реакцией, в зависимости от типа движения, выполняемого во время прерывания. Каждому прерыванию присваивается идентификатор прерывания. Он получает свою идентичность путем создания переменной (типа данных «intnum») и подключения ее к процедуре прерывания «TRAP routine». Идентификатор прерывания (переменная) затем используется для установки прерывания, то есть для указания причины прерывания. Это может быть одно из следующих событий:

— Вход или выход устанавливается в единицу или в ноль.

— После установки прерывания проходит определенное количество времени.

— Достигнута конкретная позиция.

CONNECT: инструкция для установки прерывания и привязки к определенной процедуре обработчику

ISignalDI: инструкция для настройки прерывания от цифрового сигнала

ITimer: инструкция для настройки прерывания по времени

IPers: инструкция для настройки прерывания по изменению переменной

IError: инструкция для настройки прерывания по ошибке

IDelete: инструкция для сброса прерывания

Пример использования прерывания приведен в Листинге 16.

Листинг 16. Объявление и использование прерывания.

```
VAR intnum getSample;
CONNECT getSample WITH getData;
ITimer 0.1, getSample;
...
IDelete getSample;
TRAP getData
  torque{globalCount} := GetMotorTorque(1);
  globalCount := globalCount + 1;
  RETURN;
ENDTRAP
```

Многие ошибки, возникающие при выполнении программы, могут быть обработаны в программе, а это означает, что выполнение программы не нужно прерывать. Ошибки могут быть любого типа: системные, так как деление на ноль, либо ошибки, инициированные самой программой, например некорректный штрих-код товара. Ошибка выполнения — это нештатная ситуация, связанная с выполнением определенной части программы. Ошибки идентифицируются по уникальному номеру и всегда распознаются системой. Возникновение ошибки приводит к приостановке нормального выполнения программы, и управление передается обработчику ошибок. Концепция обработчиков ошибок позволяет реагировать на ошибки, возникающие во время выполнения программы, и, возможно, восстанавливаться (продолжать работу) после них. Если дальнейшее выполнение невозможно, обработчик ошибок может, по крайней мере, гарантировать корректное завершение программы. При возникновении ошибки вызывается обработчик ошибок подпрограммы (если он есть). Также возможно создать ошибку внутри программы, а затем перейти к обработчику ошибок. В обработчике ошибок ошибки можно обрабатывать с помощью обычных инструкций. Системная

переменная «ERRNO» используется для определения типа возникшей ошибки. Возврат из обработчика ошибок может осуществляться различными способами: «RETURN», «RETRY», «TRYNEXT» и «RAISE». Если текущая процедура не имеет обработчика ошибок, внутренний обработчик ошибок робота берет на себя работу напрямую. Внутренний обработчик ошибок выдает сообщение об ошибке и останавливает выполнение программы, указав программный указатель на ошибочную инструкцию.

`RAISE`: Инструкция для создания ошибки и вызова обработчика ошибок

`BookErrNo`: Инструкция для записи нового номера ошибки в систему.

`RETRY`: Инструкция для повторного выполнения инструкции, вызвавшей ошибку.

`TRYNEXT`: Инструкция для выполнения инструкции, следующей за инструкцией, вызвавшей ошибку.

`RETURN`: Инструкция для возврата к процедуре, вызвавшей текущую процедуру (обработчик).

Любая процедура (функция) может включать в себя обработчик ошибок. Обработчик ошибок на самом деле является частью подпрограммы, и область видимости любых данных подпрограммы также включает обработчик ошибок подпрограммы. Если во время выполнения процедуры возникает ошибка, управление передается ее обработчику ошибок.

Пример использования обработчика ошибок приведен в Листинге 17.

Листинг 17. Использование обработчика ошибок.

```
FUNC num safediv( num x, num y)
  RETURN x / y;
ERROR
  IF ERRNO = ERR_DIVZERO THEN
      TPWrite "The number cannot be equal to 0";
      RETURN x;
  ENDIF
```

```
ENDFUNC


CONST errnum escape := 10;
PROC main()
  RAISE escape;
ERROR
  IF ERRNO = escape THEN
      RETURN val2;
  ENDIF
ENDPROC
```

Системная переменная «ERRNO» содержит номер (самой последней) ошибки и может использоваться обработчиком ошибок для идентификации этой ошибки. После выполнения всех необходимых действий обработчик ошибок может:

- Возобновить выполнение, начиная с инструкции, в которой произошла ошибка. Это делается с помощью инструкции «RETRY».

- Возобновить выполнение, начиная с инструкции, следующей за инструкцией, в которой произошла ошибка. Это делается с помощью инструкции «TRYNEXT».

- Вернуть управление вызывающей программе с помощью инструкции RETURN. Если подпрограмма является функцией, инструкция RETURN должна указывать подходящее возвращаемое значение.

- Передать ошибку вызывающей стороне процедуры с помощью инструкции RAISE.

6. Требования к отчету и критерии оценивания

> *В рамках выполнения практического задания, которому посвящено данное пособие, предполагается подготовка отчета о проделанной работе.*

Отчёт должен содержать следующие разделы:

1. Введение (Introduction): цель работы, постановка задачи, краткая связь с теорией.

2. Методы (Methods): станция RobotStudio, настройки tooldata/wobjdata, траектории, алгоритмы и команды RAPID.

3. Результаты (Results): скриншоты/видео, ключевые измерения: точность, повторяемость, время, отсутствие коллизий.

4. Обсуждение (Discussion): объяснение результатов, причины ошибок, влияние зон/скоростей, ограничения.

5. Выводы (Conclusion): краткий итог, предложения по улучшению.

В рамках этой части проводится сопоставление поведения виртуальной модели робототехнической ячейки в симуляции с движением реального манипулятора и обсуждаются следующие выводы:

Во-первых, переход от симуляции к реальному оборудованию неизбежно сопровождается погрешностями координат, что требует калибровки рабочего объекта перед полноценным запуском. Даже после калибровки сохраняется потребность в податливом (подпружинённом) инструменте.

Во-вторых, необходимо учитывать нелинейности реального мира: шероховатость и неплоскостность поверхности стола, прогиб листа бумаги и прочие динамические факторы, отсутствующие в идеализированной модели. Перспективным решением является внедрение алгоритмов обратной связи по силе нажатия пера на поверхность — по аналогии с тем, как регулирует усилие человек.

6. Приложения: листинг RAPID-кода, версии проекта RobotStudio, схемы сигналов и временные диаграммы.

Глоссарий основных терминов

TCP (Tool Center Point) Характеристическая точка инструмента, положение и ориентация которой программируются в инструкциях движения

Workobject (wobj) Система координат рабочего объекта (например, стола)

robtarget Структура данных целевой точки в RAPID, включающая координаты, кватернион, конфигурацию и внешние оси

MoveJ Движение по сочленениям (нелинейное, быстрое)

MoveL Линейное движение TCP в декартовом пространстве

MoveC Движение по дуге окружности

Zone (fine, z10, z100) Параметр инструкции для движения (**MoveL или MoveJ**), характеризующийся зоной сглаживания траектории (fine — точная остановка, z — радиус сглаживания в мм)

Quaternion Представление ориентации в виде четырёх чисел, устойчивое к «шарнирному замку»

confdataВектор конфигурации осей робота, определяющий одно из решений обратной задачи кинематики

Singularity Особая конфигурация манипулятора, при которой теряется степень подвижности

RAPID Язык программирования роботов ABB

RobotWare Системное ПО контроллера ABB

FlexPendant Пульт ручного управления роботом

Virtual Controller Виртуальный контроллер в RobotStudio, исполняющий RobotWare

Collision Set Набор объектов для проверки коллизий

TCP Trace Инструмент визуализации траектории TCP в RobotStudio

Контрольные вопросы

1. Что такое виртуальный контроллер в RobotStudio и почему его поведение соответствует реальному роботу?
2. Перечислите системы координат, используемые при программировании робота. В чём назначение Tool Center Point (TCP)?
3. Почему задание точек траектории относительно рабочего объекта стола предпочтительнее их задания относительно wobj0?
4. В чём различие между инструкциями движения MoveJ и MoveL? В каких случаях применяется каждая из них?
5. Что такое сингулярность манипулятора и как выбор типа движения помогает её избежать?
6. Каков смысл параметров скорости (v) и зоны (fine, z100) в инструкциях движения?
7. Из каких составляющих состоит структура данных robtarget? Для чего применяется кватернион и вектор конфигурации?
8. Опишите метод создания рабочего объекта по трём точкам. Какие оси задаются точками?
9. Зачем при переносе программы на реального робота требуется калибровка рабочего объекта?
10. Какую роль играет подпружинённый карандаш и какие нелинейности реального мира оно компенсирует?

Список литературы

1. ГОСТ Р 60.0.0.4-2019/ИСО 8373:2012: Национальный стандарт Российской Федерации. Роботы и робототехнические устройства. Термины и определения (Robotsandroboticdevices. Termsanddefinitions).
2. Е.И. Юревич, Основы робототехники. --- СПб. БЧВ-Петербург, 2005S. M. LaValle, Planningalgorithms, CambridgeUniversityPress
3. Дж. Дж. Крейг, Введение в робототехнику: механика и управление, Институт компьютерных исследований, 2013
4. Springer Handbook of Robotics, edited by B. Siciliano and O. Khatib, 2008
5. L. Freidovich, Control methods for robotic applications: lecture notes, St. Petersburg, Russia: Saint Petersburg National Research University of Information Technologies Mechanics and Optics, 2013.
6. Руководство по использованию «RobotStudio 6.04»: https://docplayer.ru/63064026-Rukovodstvo-po-ispolzovaniyu-robot-studio-6-04.html