Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Удмуртский государственный университет»
Институт математики, информационных технологий и физики
Кафедра вычислительной механики

Программирование в образовательном робототехническом комплексе FANUC

Учебно-методическое пособие

Ижевск
2023

УДК 004.896(075.8)
ББК 32.973.4я73
П784

Рекомендовано к изданию Учебно-методическим советом УдГУ

Рецензенты: канд. физ-мат. наук, доцент, директор ИМИТИФ ФГБОУ ВО «УдГУ» Л.Е. Тонков,
д-р физ.-мат. наук, доцент, профессор каф. тепловых двигателей и установок «ИжГТУ» Е.В. Ветчанин.

Составитель: Шкляев К.Я.

П784 Программирование в образовательном робототехническом комплексе FANUC : учеб.-метод. пособие / сост. К.Я. Шкляев. – Ижевск : Удмуртский университет, 2023. – 73 с.

ISBN 978-5-4312-1147-8

Учебно-методическое пособие сдержит описание принципов работы FANUC education cell, инструкции по работе, примеры программ и задания для практического выполнения.

Пособите предназначено для использования в рамках изучения дисциплин, связанных с САПР и робототехникой.

УДК 004.896(075.8)
ББК 32.973.4я73

ISBN 978-5-4312-1147-8

Содержание

1 ВВЕДЕНИЕ

Область, объединяющая науку, технику и технологии – это робототехника. В ней проводятся исследования с целью создания машин, которые смогут повторять действия человека и даже во многом его заменить. В робототехнику входят проектирование, создание и программирование роботов. Программное обеспечение роботов представляет собой совокупность программ и связанных с ними данных, которые разрабатываются, производятся, продаются и сопровождаются для обеспечения функционирования промышленных роботов. Программирование - завершающий этап создания робота. Полностью собранный высокотехнологичный робот фактически является только макетом до загрузки программы, которая научит его совершать определенные операции и исполнять команды, определенные оператором.

Роботизированный комплекс FANUC education cell предназначен для развития навыков программирования промышленных роботов и станков с ЧПУ (числовым программным управлением). Подобное оборудование является неотъемлемой частью любого современного, высокотехнологичного производства. Основными преимуществами робототехники и систем с ЧПУ являются:

- Высокая степень автоматизации, сводящая к минимуму взаимодействие оператора и оборудования, а в некоторых случаях системы с ЧПУ могут работать автономно, без участия человека. Что повышает производительность труда, позволяя одному рабочему обслуживать несколько станков.

- Точность исполнения всех операций и предсказуемое качество выходной продукции. В отличие от человека робот может выполнять одну и ту же операцию раз за разом абсолютно

одинаково. Поэтому единожды написанная и отлаженная программа будет всегда выдавать идентичные детали за одинаковое время.

- Гибкость производства. Единственный станок с ЧПУ может изготавливать множество различных деталей. Всё что для этого нужно – заменить программу и инструмент робота. Это позволяет в кратчайшие сроки внедрять новые детали в производство и переводить станки между различными линиями.

При этом общий принцип работы FANUC education cell довольно прост: контроллер робота перемещает последнего от одной точки к другой по заданной траектории, согласно программе. Дальнейшее пособие посвящено разработке программ, управляющих роботом.

Программы для управления промышленными роботами разрабатываются двумя способами – онлайн и оффлайн. Студенты при изучении робототехники должны овладеть умениями программирования роботов непосредственно на месте их установки, а также в редакторах, поставляемых с комплексно программного обеспечения и загрузки текстовая программы в компьютер робота, с последующей отладкой и корректировкой.

Роль программного обеспечения промышленных роботов можно рассматривать с трех точек зрения: специалиста по эксплуатации, разработчика роботизированной системы и производителя роботов. Требования, предъявляемые к нему различными группами пользователей, зачастую противоречивы. При изучении данного пособия, студент получает знания и навыки инженера-программиста, занимающегося роботизацией и располагающего средствами разработки программного

обеспечения и сервисными программами, которые позволяют ему выполнять свою работу быстро и эффективно.

Материал, рассмотренный в данном пособии, особенно актуален в сложившейся ситуации, когда многие производители роботов отказались от предоставления услуг и поддержки, а издания с учебно-методическим обеспечением проведения занятий, ориентированные на современные подходы к программированию роботов, практически полностью отсутствуют.

Цель данного пособия – обеспечить методическую поддержку для приобретения студентами практических навыков составления и выполнения программ для промышленных роботов на одном из роботоориентированных языков программирования.

Первым обязательным этапам изучения робототехники является знакомством с устройством робототехнической системы в целом, управлением, режимами работы и базовыми командами. Во второй части последовательно рассматриваются режимы выбора систем координат, ручного режима управления. Следующая основная часть пособия посвящена непосредственно программированию и запуску программ на выполнение с панели оператора и пульта обучения. В заключительном разделе приводятся примеры программ с подробным описанием и использованием дополнительных средств. Для самостоятельного решения и программирования предлагаются задачи, раскрывающие весь спектр возможностей робототехнической ячейки и позволяющие оценить уровень освоения изучаемого материала.

Отличительной особенностью пособия является содержание практических заданий, каждое из которых сопровождается рекомендациями и теоретическими сведениями из области программирования промышленных роботов. Выполнение учебных

заданий формирует умения и навыки по геометрическому моделированию, программированию промышленных роботов. Предлагаются задания для тестирования начальных умений по программированию роботов.

Многие вопросы, затронутые в пособии, активно обсуждались с доктором наук, профессором Кафедры вычислительной механики ИМИТиФ С.П. Копысовым. Рукопись пособия им также была внимательно прочитана. Автор с благодарностью учел все замечания.

2 НАЧАЛО РАБОТЫ

Роботизированная система FANUC (рисунок 1) включает в себя такие элементы, как непосредственно манипулятор робота, контроллер, пульт Teach Pendant, программное обеспечение.

Рис. 1: Робототехническая система FANUC

2.1 Робот

Робот состоит из нескольких плечей, соединённых последовательно между собой, точка соединения плечей — это **шарнир** или **ось**. Вся система приводится в движение сервомоторами. В нашем случае робот имеет шесть осей: **J1**, **J2**, **J3**, **J4**, **J5**, **J6** (рисунок 2).

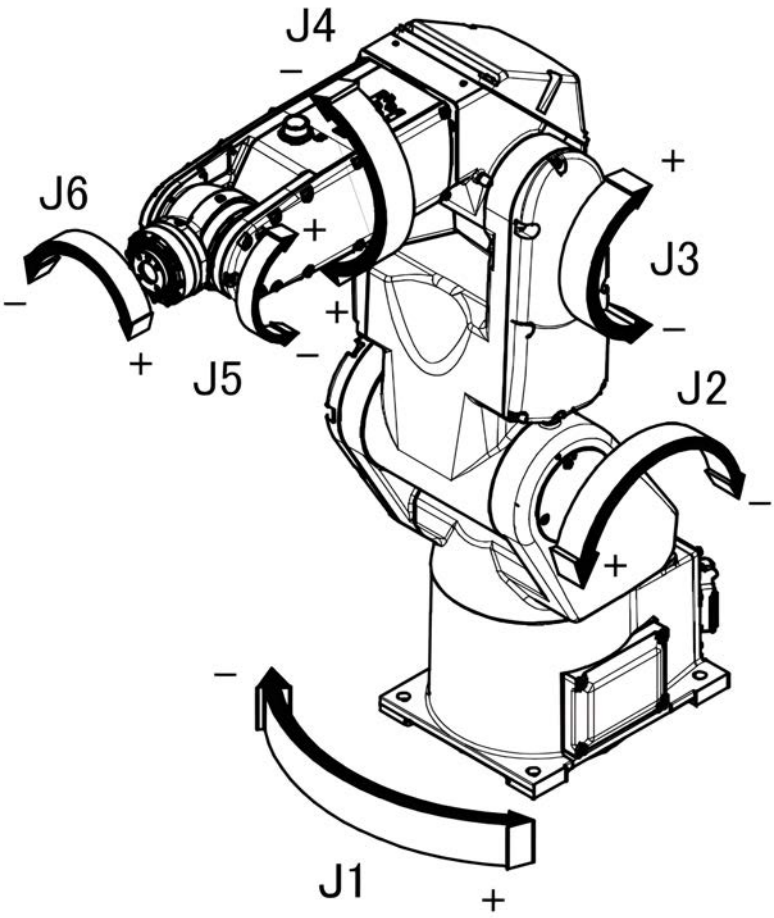

Рис. 2: Кинематическая схема робота

Управление осуществляется посредством ручного ввода команд или задания алгоритма работы, называемого программой. Основными органами управления роботом являются панель оператора (рисунок 3) и пульт обучения (рисунок 6), о них речь пойдёт далее.

Рис. 3: Панель оператора

Включить питание контроллера и робота можно при помощи переключателя на передней панели, повернув его по часовой стрелке в положение **ON** (рисунок 4).Выключение происходит при переводе переключателя в положение **OFF**. Для открытия дверцы контроллера необходимо перевести переключатель в положение **OPEN**.

Рис. 4: Выключатель питания контроллера

2.2 Режимы работы контроллера

Робот имеет три основных режима работы: **AUTO**, **T1** и **T2**. Выбор режима происходит по средствам **трёхпозиционного переключателя**, данный переключатель расположен на передней панели контроллера (рисунок 5).

Рис. 5: Трёхпозиционный переключатель

2.2.1 Режим тестирования T1

Тестовый режим **T1** используется для написания программ и их первичной отладки. Скорость движения **TCP** (вершина инструмента, подробнее в разделе 3.1.4) ограниченна значением в **250 мм/сек**. Такое ограничение скорости делает невозможным проверку фактической траектории движения.

Помимо написания кода и отладки, данный режим можно использовать для ручного перемещения робота с помощью пульта Teach Pendant. Для работы в данном режиме необходимо:

1. перевести **трёхпозиционный переключатель** в верхнее положение **T1** при помощи ключа;

2. включить пуль Teach Pendant (об устройстве пульта обучения подробнее в разделе 2.3).

2.2.2 Режим тестирования T2

Тестовый режим **T2** предназначен для окончательной отладки программы, ограничение скорости при выполнении программ отсутствует. В ручном режиме скорость ограничена значением **250 мм/сек**.

Для работы в данном режиме необходимо:

1. перевести **трёхпозиционный переключатель** в правое положение **T2** при помощи ключа;

2. включить пуль Teach Pendant (об устройстве пульта обучения подробнее в разделе 2.3).

2.2.3 AUTO (автоматический)

Режим **AUTO** предназначен для выполнения отлаженных программ. В данном режиме отсутствует ограничение скорости и невозможно управление роботом вручную. Программа может быть запущенна только с панели оператора или внешних устройств.

2.3 Пульт обучения

Пульт обучения (рисунок 6) позволяет осуществлять управление и настройку робота, разрабатывать и отлаживать

программы управления роботом. Ниже приведён список элементов пульта.

- Дисплей, отображающий программные меню **SpotTool+**.
- Клавиши.
- **Выключатель пульта обучения**, верхний левый угол.
- **Переключатель аварийной блокировки**, находится с обратной стороны пульта. Перемещение робота возможно только при нажатом переключателе (сильное нажатие также приводит к невозможности использование робота).
- **Кнопка аварийной остановки**, верхний правый угол. Мгновенно останавливает робота вне зависимости от режима работы или выполняемой программы. Для того, что бы выйти из режима аварийной остановки необходимо повернуть кнопку по часовой стрелке.

Рис. 6: Пульт обучения

Клавиши пульта обучения (рисунок 6):

1. Клавиша **FCTN**: выводит на экран список функций;
2. Клавиши для работы с программами:
   (a) **SELECT** (выбор) – отображает список зарегистрированных программ;
   (b) **EDIT** (редактировать) – перенаправляет на экран редактирования программы;
   (c) **DATA** (данные) – экран данных регистров, регистров положения, и так далее;

3. Клавиша **HOLD**: немедленно останавливает выполнение программы;

4. Клавиша **FWD** (вперед): позволяет начать выполнять программу или, в процессе выполнения, перейти к следующей инструкции;

5. Клавиша **BWD** (назад): переход к предыдущей инструкции программы;

6. Клавиша **COORD** (координата): выбор системы координат в режимах **T1** и **T2**;

7. Клавиши перемещения робота в выбранной системе координат (о системах координат речь пойдёт далее):

   (a) **+X (J1)**, **-X (J1)** – перемещение инструмента вдоль оси **Ox** (вращение первого шарнира);

   (b) **+Y (J2)**, **-Y (J2)** – перемещение инструмента вдоль оси **Oy** (вращение второго шарнира);

   (c) **+Z (J3)**, **-Z (J3)** – перемещение инструмента вдоль оси **Oz** (вращение третьего шарнира);

   (d) $\overleftarrow{\textbf{+X (J4)}}$, $\overrightarrow{\textbf{-X (J4)}}$ – вращение инструмента вдоль оси **Ox** (вращение четвертого шарнира);

   (e) $\overleftarrow{\textbf{+Y (J5)}}$, $\overrightarrow{\textbf{-Y (J5)}}$ – вращение инструмента вдоль оси **Oy** (вращение пятого шарнира);

   (f) $\overleftarrow{\textbf{+Z (J6)}}$, $\overrightarrow{\textbf{-Z (J6)}}$ – вращение инструмента вдоль оси **Oz** (вращение шестого шарнира).

8. **+%**, **-%** – клавиши регулирования скорости при ручном управлении;

9. Клавиша **ENTER** – ввод;

10. Клавиша **BACK SPACE**: удаление текста, используется при вводе названий и описаний;

11. Клавиша **RESET**: обновление состояний системы, нажмите эту клавишу после устранения причин ошибок;

12. Клавиша **STEP**: переключение между пошаговым и непрерывным режимом работы;

13. **Клавиши со стрелками** – перемещение курсора;

14. Клавиша **MENU**: вызов экранного меню;

15. Клавиша **SHIFT**: используется в сочетании с выделенными синим клавишами. Левая и правая клавиши идентичны;

16. Клавиши **F1**, **F2**, **F3**, **F4**, **F5** – набор функциональных клавиш, назначение каждой клавиши подписанно в нижней части экрана над соответствующей клавишей;

17. **NEXT** и **PREV** – переключение вкладок функциональных клавиш;

2.4 Базовые команды

Порядок включения

1. Расставить используемые объекты внутри рабочего пространства, неиспользуемые объекты убрать;

2. Если в процессе работы используется система iR Vision, то необходимо снять защитную крышку с объектива камеры;

3. Закрыть дверцу, повернуть фиксатор;

4. Выбрать режим работы робота при помощи трёхпозиционного переключателя, в зависимости от режима работы включите или выключите пульт обучения (подробней в разделе 2.2);

5. Повернуть выключатель (рисунок 4) по часовой стрелке в красную зону.

Порядок выключения

1. Закончить работу;

2. Перевести робота в домашнее положение (см. ниже);

3. Повернуть переключатель (рисунок 4) против часовой стрелки в зелёную зону;

4. Навести порядок в рабочем пространстве;

5. Надеть защитную крышку на объектив камеры;

6. Закрыть дверцу, повернуть фиксатор;

7. Повесить пуль Teach Pendant на специальный держатель.

Возврат робота в домашнее положение

1. Нажать клавишу **SELECT**;

2. В появившемся меню выбрать программу **AA_HOME**;

3. Запустить программу с помощью пульта обучения или контроллера робота.

Аварийная остановка В случае, если необходимо срочно остановить робота, то нажмите кнопку аварийной остановки на пульте обучения или на контроллере робота.

Для того что бы продолжить работу после аварийной остановки поверните нажатую кнопку по часовой стрелке, затем нажмите клавишу **RESET** на пульте обучения.

3 РУЧНОЙ РЕЖИМ УПРАВЛЕНИЯ

В данном режиме робот перемещается по команде пользователя.

В ручном режиме пользователем задаются скорость перемещения и система координат.

Скорость робота задаётся клавишами +% и -% (рисунок 7).

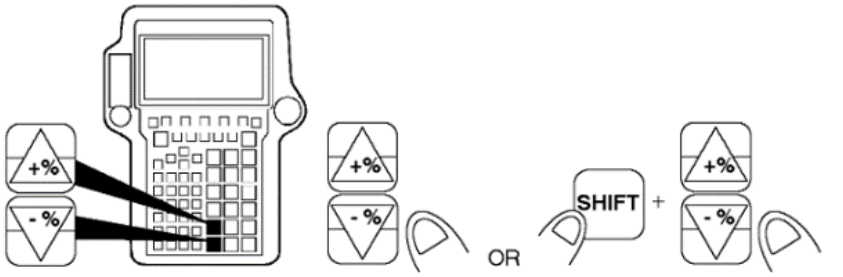

Рис. 7: Задание скорости подачи

В верхнем правом углу пользователь может увидеть текущее значение в процентах от максимально возможной. Шаг изменения скорости в пределах от **1%** до **5%** - **1%**, в пределах от **5%** до **100%** - **5%**. Также существуют две скорости для точной подгонки и очень точной подгонки - **FINE** и **VFINE** соответственно (рисунок 8).

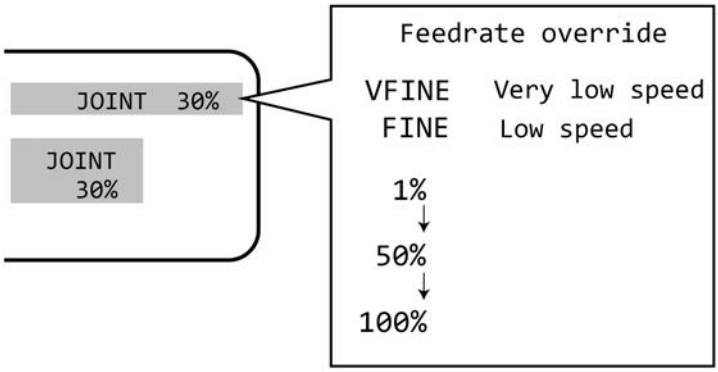

Рис. 8: Вид экрана изменения скорости подачи

3.1 Системы координат при ручном управлении

Системы координат (далее СК) определяет то, как именно робот будет реагировать на нажатие клавиш перемещения (раздел 2.3) и то по какой траектории будет двигаться **TCP**. TCP, оно же **Tool Center Point** – это воображаемая точка, находящаяся в вершине инструмента. Положение **TCP** относительно вершины инструмента может изменено вручную (раздел 3.1.4).

Для пользователя доступны четыре основных системы координат:

- **JOINT**;
- **WORLD**;
- **USER**;
- **TOOL**.

Подробнее об этих СК речь пойдёт дальше.

Текущая система координат отображается в верхнем правом углу экрана пульта обучения. Выбор системы координат осуществляется при помощи клавиши **COORD**. (рисунок 9).

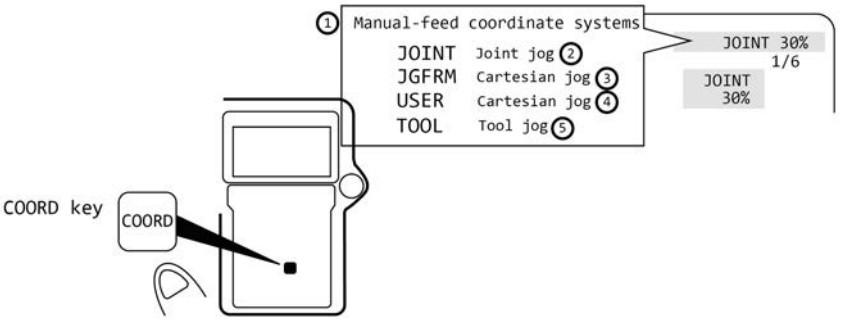

Рис. 9: Выбор системы координат

3.1.1 Система координат шарнира JOINT

Данная система координат позволяет управлять движением шарниров напрямую, независимо друг от друга, согласно рисунку 2. Каждая пара клавиш перемещения робота активирует свой шарнир (сустав), не затрагивая остальные (шарнир **J2** является исключением, его движение влияет на **J3**, но не наоборот). При использовании СК **JOINT** рекомендуется настраивать положения суставов последовательно, от первого к шестому.

Траектория **TCP** представляет собой сложную, нелинейную кривую, состоящую из дуг окружностей.

3.1.2 Мировая система координат WORLD

В данной система координат перемещение робота происходит вдоль осей мировой ПДСК (рисунок 10). Клавиши перемещения

позволяют перемещать **TCP** строго по прямой вдоль осей oX, oY, oZ и совершать повороты вокруг **TCP**.

В СК **WORLD TCP** двигается вдоль ломаной кривой, отрезки которой сонаправлены с осями координат.

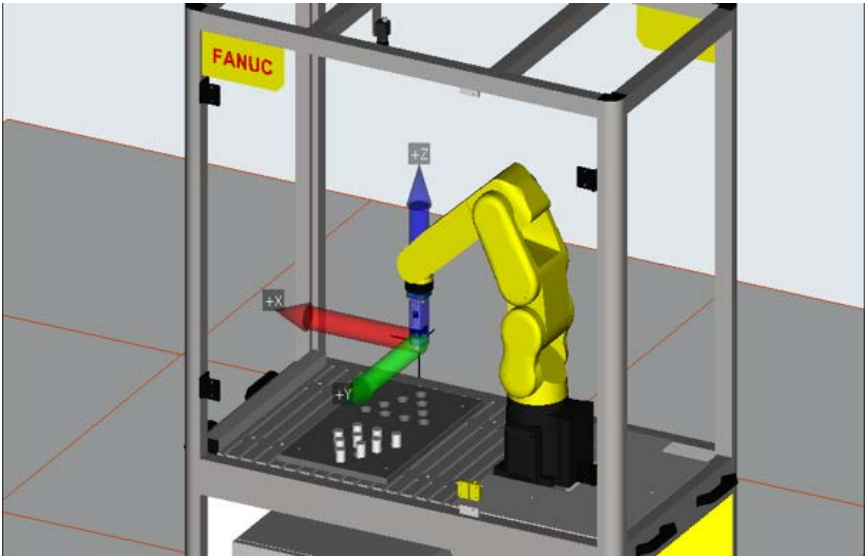

Рис. 10: Система координат **WORLD**

3.1.3 Пользовательская система координат USER

СК **USER** – аналогична СК **WORLD**, отличается направлением осей и имеет возможность редактирования пользователем. Применяет в тех случаях, когда остальные СК не удобны. Траектория **TCP** аналогична СК **WORLD**.

Для того чтобы задать пользовательскую систему координат нажмите клавишу **MENU** → **SETUP** → **Frames**. В появившемся меню нажмите клавишу **F3 [OTHER]**, и выберете пункт **User Frame**. Перед вами появится список из

девяти пользовательских систем координат, с помощью клавиш со стрелками выберете ту СК, которую хотите изменить (**не рекомендуется изменять СК №1**) и нажмите **ENTER**. В открывшемся меню напишите название вашей СК в поле **comment**, затем на пульте нажмите клавишу **F2 [METHOD]** и **клавишами со стрелками** выберете метод задания СК.

- **Three Point** (**Метод трёх точек**). Суть метода состоит в записи трёх опорных точек декартовой системы координат. Для задания СК необходимо:

  1. Переместите робота в начало новой системы координат. С помощью клавиш со стрелками выберете пункт **Orient Origin Point**, затем нажмите сочетание клавиш **SHIFT** + **F5**, в результате, программа запомнит положение робота, а графа напротив поля **Orient Origin Point**, изменит статус с **UNINIT**, на **RECORDED**;
  2. С помощью графы **X Direction Point** задаётся направление оси **Ox**. Переместите робота вдоль оси **Ox**, и в поле **X Direction Point** запишите его положение;
  3. В графе **Y Direction Point** задайте последнюю точку, аналогично первым двум пунктам. Эта точка, совместно с первыми двумя, задаёт положение плоскости **Oxy**, а также положительное направление оси **Oy**. Ось **Oz** определяется автоматически так, чтобы итоговая система координат была правой;
  4. Если всё сделано правильно, то напротив каждого пункта меню будет появиться надпись **USED**.

- **Four Point** (**Метод четырех точек**). Суть в следующем: первые две точки задают направление оси **Ox**, третья точка

задаёт плоскость **Oxy**, четвертая – начало координат. Для задания СК необходимо:

1. Повторить пункты 1-3 из **метода трёх точек**.
2. Переместить робота в точку, в которой должно находиться начало координат. Выбрать пункт **System Origin** и нажать комбинацию **SHIFT** + **F5**;
3. Если всё сделано правильно, то напротив каждого пункта меню будет появиться надпись **USED**.

- **Direct Entry** (**Прямой ввод**). Данный метод используется для прямого ввода ориентации и СК **USER** относительно системы координат **WORLD**.

3.1.4 Система координат TOOL

Данная декартова система координат привязана непосредственно к инструменту робота (в отличие от СК **WOLRD**, которая сохраняет ориентацию при повороте инструмента, СК **TOOL** вращается совместно с инструментом) и может быть изменена пользователем. Центром данной системы координат является **TCP**.

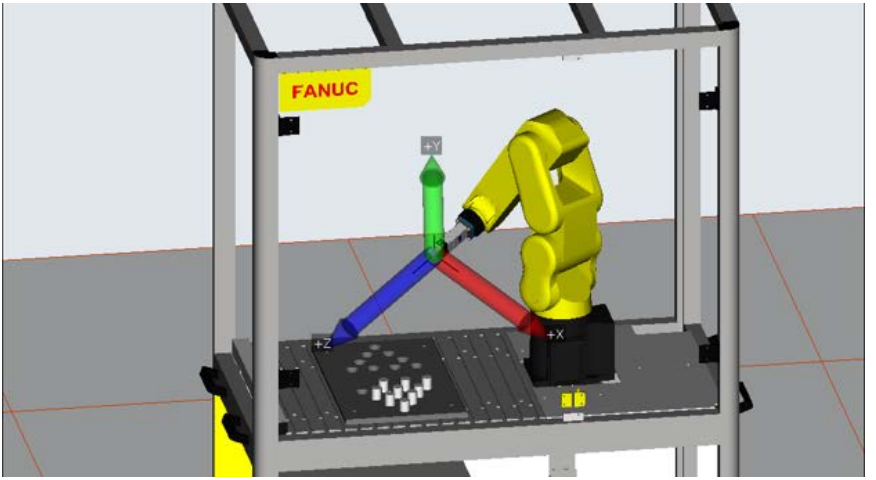

Рис. 11: Система координат **TOOL** (СК сохранила ориентацию инструмента при повороте)

Для записи пользовательской СК выполните следующие действия: **MENU** → **SETUP** → **Frames**. В появившемся меню нажмите клавишу **F3 [OTHER]**, и выберете пункт **Tool Frame**. Перед вами появится список из девяти пользовательских систем координат, с помощью клавиш со стрелками выберете ту СК, которую хотите изменить (**не рекомендуется изменять СК №1**) и нажмите **ENTER**. В открывшемся меню напишите название вашей СК в поле **comment**, затем на пульте нажмите клавишу **F2 [METHOD]** и клавишами со стрелками выберете метод задания СК.

- **Three Point** (**Метод трех точек**). Суть метода состоит в изменении **TCP**, с помощью записи трёх положений робота в одной и той же точке в различных ориентациях.

1. Переместите вершину инструмента в какую-нибудь точку и запишите её положение клавишами **SHIFT** + **F3**;
2. Поверните вершину инструмента вокруг одной из осей ПДСК, и снова верните его в первую точку, затем запишите получившееся положение;
3. Повторите пункт 2 для другой оси ПДСК.

- **Six Point** (**Метод шести точек**). Данный метод позволяет не только задать положение **TCP**, но и изменить ориентацию СК.

  1. Для задания **TCP** повторите пункты 1-3 из **метода трёх точек**;
  2. Чтобы задать направление осей, расположите робота так, чтобы ось Oz новой системы координат была направленна вертикально;
  3. Для удобства переключитесь на мировую систему координат (**WORLD**), переместите робота вдоль оси Ox новой системы координат и запишите новое положение в поле **X Direction Point**;
  4. Переместите робота вдоль оси **Oz** новой системы координат и запишите новое положение в поле **Z Direction Point**.

3.2 Перемещение робота в ручном режиме

Для перемещения робота как в ручном, так и в режиме выполнения программ необходимым условием является отсутствие ошибок. В случае ошибки на пульте управления горит индикатор **FAULT**, а на экране отображается код ошибки (раздел 3.3).

Что бы непосредственно управлять роботом в ручном режиме необходимо:

- удерживать переключатель аварийной остановки в среднем положении;
- удерживать клавишу **SHIFT**;
- нажать клавишу перемещения (рисунок 12).

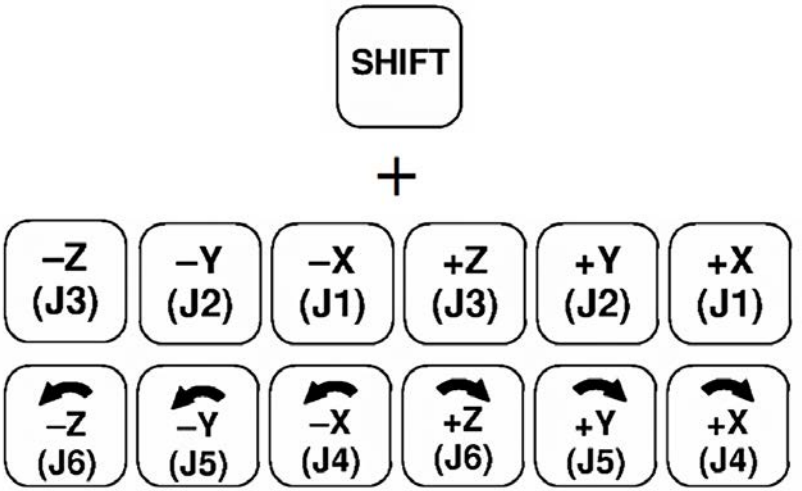

Рис. 12: Клавиши перемещения при ручном управлении

Что бы отобразить на экране текущие координаты робота: **MENU** → **NEXT** → **4D GRAPHICS** → **Position Display**.

3.3 Коды ошибок

Код ошибки представляет собой шифр, который позволяет быстро определить причину отказа. В случае возникновения в системе какой-либо неполадки соответствующий код отображается

в верней части экрана, помимо этого на пульте загорается индикатор **FAULT**.

В таблице ниже собраны наиболее частые причины неполадок.

Код ошибки	Описание
SRVO-001	Нажата кнопка аварийной остановки на панели оператора
SRVO-002	Нажата кнопка аварийной остановки на пульте обучения
SRVO-003	Отпущен переключатель аварийной блокировки
SRVO-004	Открыта рабочая камера
SRVO-005	Робот нарушил разрешенный диапазон движения
SRVO-006	Неисправность инструмента
SERVO-105	Открыт контроллер робота

Устранив причину ошибки, для продолжения дальнейшей работы необходимо нажать **RESET**, при этом, режимах **T1** и **T2** удержание **переключателя аварийной блокировки** является обязательным.

4 ПРОГРАММИРОВАНИЕ РОБОТА

Что бы робот мог действовать самостоятельно, без помощи оператора, ему необходима программа. Программа представляет собой алгоритм действий, записанный в понятной для контроллера робота форме. Этот раздел познакомит читателя с методами написания и редактирования программ.

4.1 Регистрация программы с помощью пульта обучения

Прежде чем перейти к написанию первой программы её необходимо зарегистрировать. Для этого выполните следующие шаги:

1. Включите робота согласно инструкции, выберите режим **T1**;
2. Перейдите на экран выбора программ с помощью клавиши **SELECT**;
3. Нажатием клавиши **F2 [CREATE]**;
4. В подменю **Alpha input** выбрать **options/Keybd**;
5. Нажать **F5 [KEYBOARD]**;
6. Ввести имя программы;
7. Дополнительные данные о программе можно ввести при помощи клавиши **F2 [DETAIL]** (не обязательно);
8. После ввода всех параметров нажмите **F3 [EDIT]**, вы попадёте на экран редактирования программы.

4.2 Регистры

Регистр – это переменная, которая позволяет хранить некоторые данные, тип хранимых данных зависит от типа регистров. Количество регистров в системе ограниченно. Некоторые из регистров уже содержат данные и/или комментарий, крайне не рекомендуется изменять их значение.

Регистры данных хранят числовые значения, которыми могут быть:

- Константа;
- Значение элемента регистра положения;
- Аналоговый выходной сигнал;
- Цифровой входной сигнал системы;
- Цифровой выходной сигнал системы;
- Цифровой входной сигнал робота;
- Цифровой выходной сигнал робота;
- Входной сигнал периферийного устройства;
- Выходной сигнал периферийного устройства.

Обратиться к регистру можно командой **R[i]**, где **R** обозначает регистр данных, а **i** его номер в системе. Всего в системе 200 регистров.

Над регистрами данны можно проводить арифметические операции. Для этого:

1. Переместите курсор на номер строки, в которую хотите добавить команду регистра;

2. Нажмите клавишу **F1 [INST]**. Откроется меню команды управления (рисунок 13). Если вместо **F1 [INST]** на экране отображается **F1 [POINT]**, то нажмите **NEXT**;

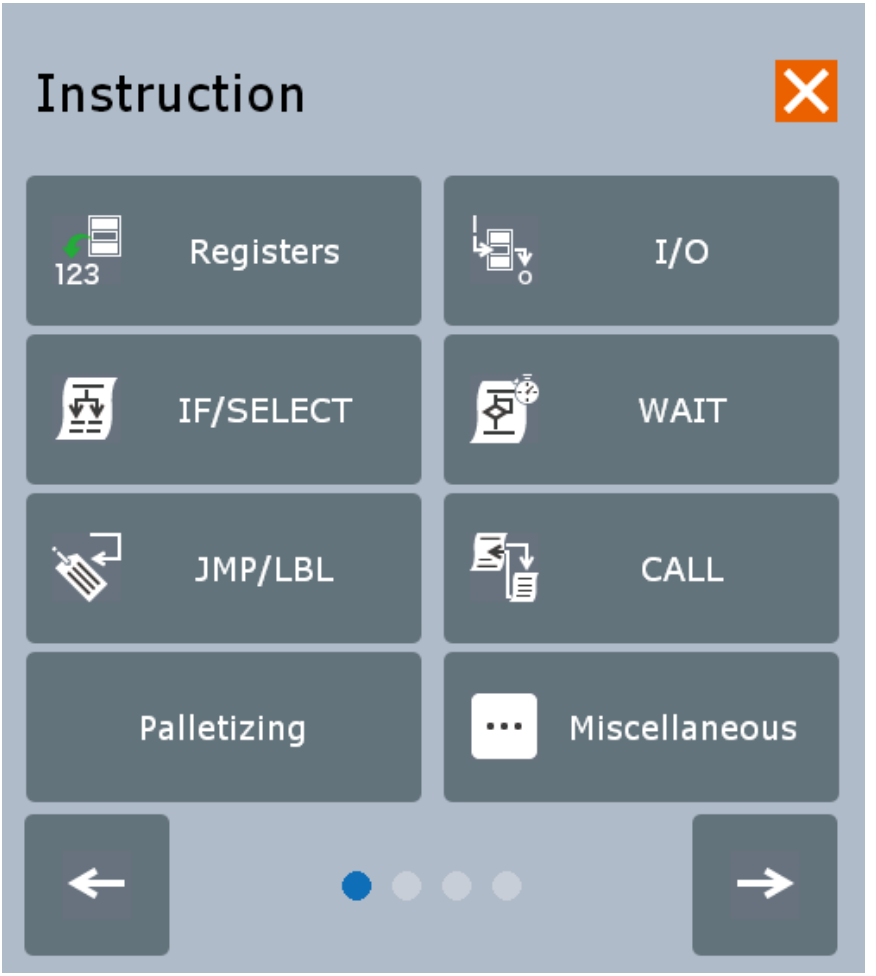

Рис. 13: Меню команды управления

3. Выберите **REGISTERS**;

4. Выберите необходимое действие с регистром (рисунок 14);

Рис. 14: Виды команд регистра

5. после выбора команды вместо многоточий впишите необходимые регистры.

Регистры положения содержат в себе набор чисел, позволяющий однозначно описать положение робота. Легко видеть, что для шести-осевого робота достаточно шести чисел.

Обращение к осуществляется командой **PR[i]**, где **PR** – регистр положение, а **i** это номер регистра в системе. Позицию конкретного шарнира можно получить командой **PR[i, j]**, здесь **i** – номер регистра, **j** – номер сустава. Всего в системе 100 регистров положения.

Настройка регистров положения.

1. Нажмите клавишу **MENU**, чтобы отобразить экранное меню;
2. Нажмите клавиши **DATA** → **F1 [TYPE]** → **Position Reg**;
3. Откроется экран регистров положения (рисунок 15);

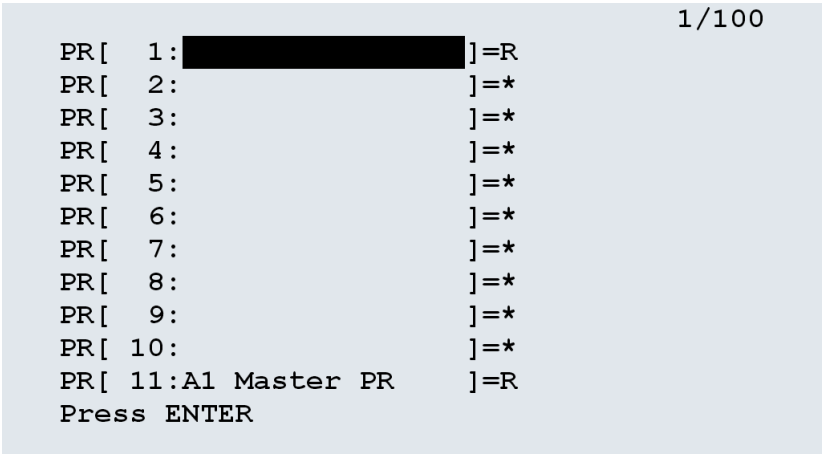

Рис. 15: Экран регистров положения

4. После номера регистра опционально можно ввести комментарий.

5. Для изменения значения регистра положения переместите курсор в поле значения регистра положения. Затем, при нажатой клавише **SHIFT**, нажмите **F3 [RECORD]**, текущее положение робота запишется в регистр;

   (a) Индикация **R** указывает, что в регистре положения уже хранится значение, введённое в режиме обучения;

   (b) Звёздочка (*) указывает, что значение не хранится (рисунок 15).

6. Для удаления данных позиционирования, загруженных в регистр положения нажмите **F5 [CLEAR]**, при нажатой клавише **SHIFT** (рисунок 16);

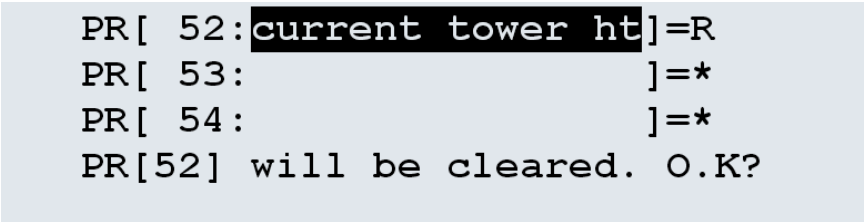

Рис. 16: Удаление регистра положения

7. Чтобы узнать текущее значение данных позиционирования, нажмите **F4 [POSITION]**. Появится экран данных позиционирования. Для изменения значения переместите курсор к нужному полю, затем введите новое значение (рисунок 17);

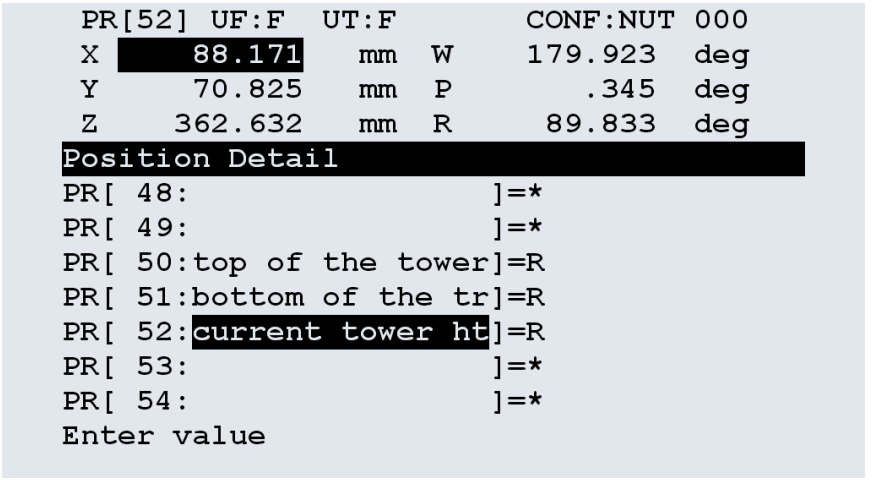

Рис. 17: Редактирование регистра

8. По окончанию настройки нажмите **F4 [DONE]**.

4.3 Команды перемещения

Команда перемещения указывает роботу как, по какой траектории и с какой скоростью перемещаться. Для корректной работы команды перемещения так или иначе должны указанны следующие её элементы:

- Формат движения;
- Данные позиционирования;
- Скорость перемещения;

4.3.1 Формат движения

Существуют три формата движения:

- Движение шарнира (**J**);
- Линейное перемещение (**L**);
- Круговое движение (**C**).

Они отличаются типом итоговой траектории.

Движение шарнира J задаёт перемещение робота из одного положения в другое одновременным поворотом всех шарниров на определённый угол. Результирующая траектория, как правило, нелинейна (рисунок 18). В процессе перемещения ориентация инструмента может изменяться. Скорость движения указывается в процентах от максимально возможной.

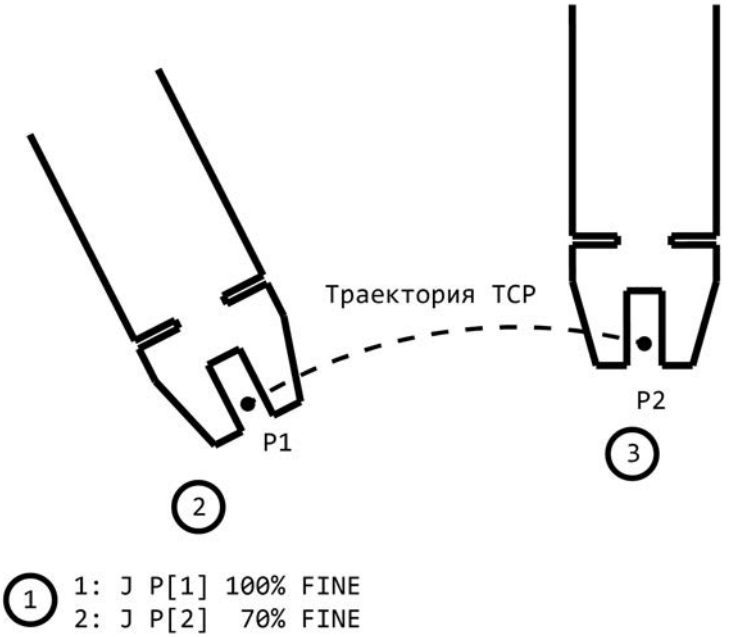

Рис. 18: Движение шарнира **J**

Режим линейного перемещения (L) , как следует из названия, перемещает **TCP** по прямой линии. Положение инструмента при этом будет так же изменяться линейно. Скорость движения указывается явно, в виде числового значения и единицы измерения, доступны следующие единицы:**мм/с**, **см/мин**, **дюймы/мин** и % от максимальной скорости.

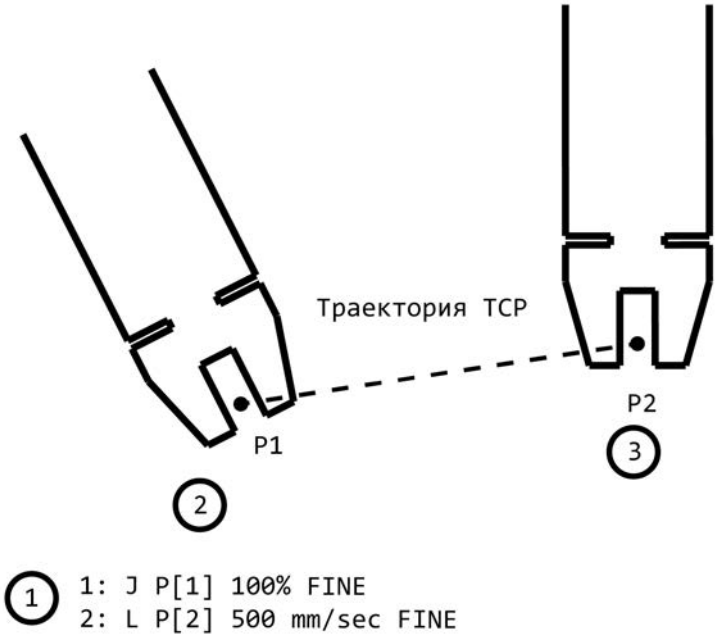

Рис. 19: Линейное перемещение (**L**)

Круговое движение (C) позволяет **TCP** двигаться по окружности, при этом инструмент сохраняет свою ориентацию относительно оси вращения. Вид траектории и формат команды можно увидеть на рисунке 20, скорость перемещения указывается аналогично линейному движению (раздел 4.3.1).

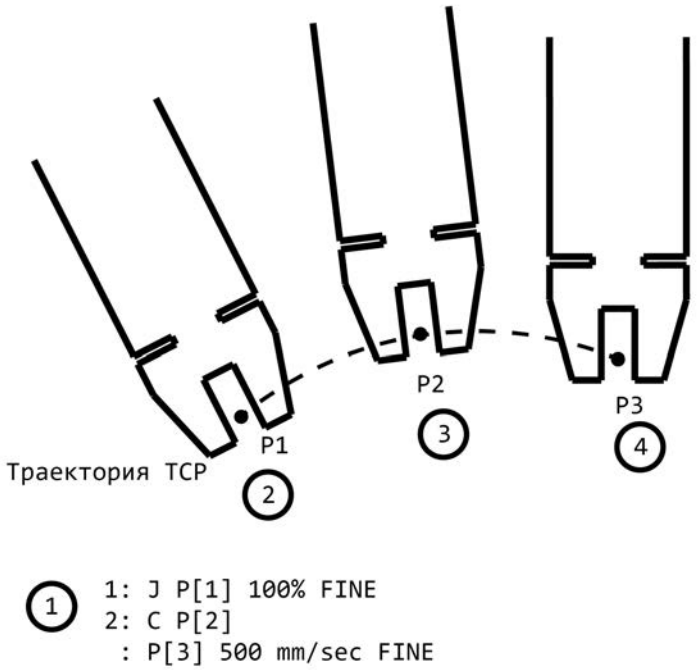

Рис. 20: Круговое движение (**C**)

4.3.2 Данные позиционирования

В команде перемещения, помимо прочего, указывается переменная положения (**P[i]**), которая хранит углы поворотов шарниров или декартовы координаты и ориентацию **TCP**. Переменная положения сохраняется в момент обучения робота. Вместо переменной положения может выступать регистр положения **PR[i]** (раздел 4.2).

Подробнее о данных позиционирования для конкретной команды можно узнать при помощи клавиши **F5 [POSITION]**

4.3.3 Тип траектории

В отдельных случаях может возникать необходимость добиваться максимальной точности позиционирования или ускорять движение робота по траектории. Такие задачи решаются при помощи траекторий различных типов. Тип траектории указывается отдельно для каждой команды перемещения, всего типов два.

Тип FINE имеет следующий формат команды:

J P[i] 50% FINE

В каждой точке траектории с пометкой **FINE** робот совершает полную останову. Данный режим перемещения используется для достижения наибольшей точности позиционирования робота.

Тип CNT. Пример использования ниже.

J P[i] 50% CNT50

При вызове команды перемещения с пометкой **CNT** робот движется непрерывно, без остановок. Также **CNT** позволяет сглаживать углы траектории, число после **CNT** отвечает за степень сглаживания от 0 – минимальной, до 100 – максимальной (рисунок 21).

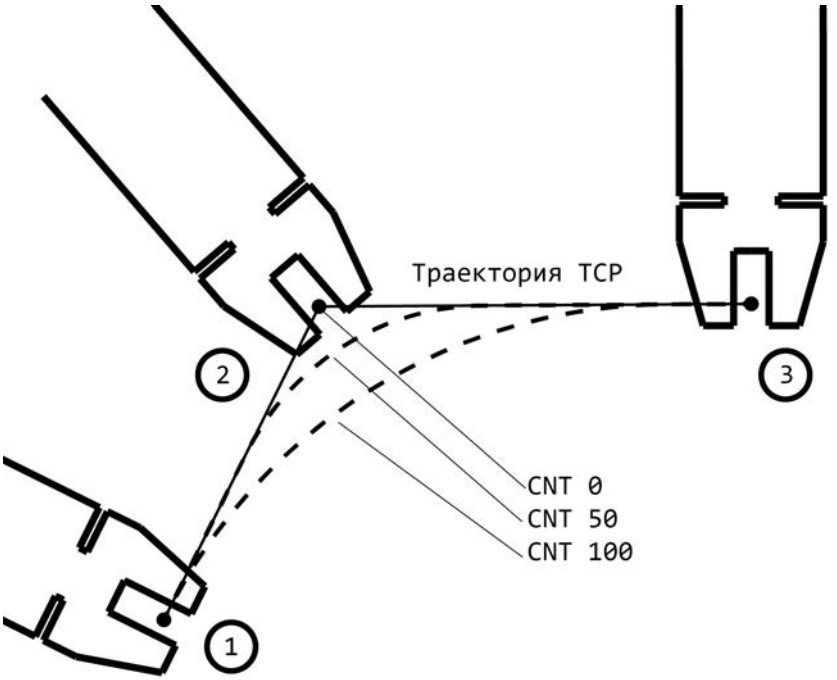

Рис. 21: Движение робота в режиме **CNT**

4.3.4 Изменение данных позиционирования с помощью робота

1. В режиме **T1** или **T2** перейдите на экран редактирования программы, которые вы хотите изменить.

2. Переместите курсор на номер строки, которую необходимо изменить;

3. Переместите робота в новое положение;

4. Одновременно нажмите **F5 [TOUCHUP]** и **SHIFT**. Новое положение записано.

4.3.5 Изменение данных позиционирования с помощью экрана информации

1. В режиме **T1** или **T2** перейдите на экран редактирования программы, которые вы хотите изменить.

2. Переместите курсор на номер строки, которую необходимо изменить;

3. Нажмите клавишу **F5 [POSITION]** для отображения экрана редактирования координат (рисунок 22);

4. На экране редактирования координат измените желаемые координаты.

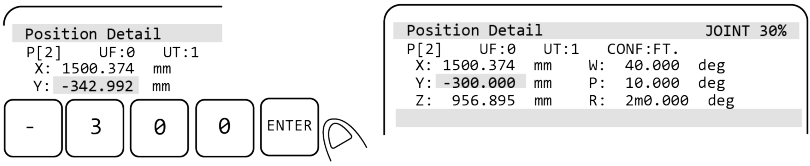

Рис. 22: Экран редактирования координат

4.3.6 Изменение команды перемещения

Для изменения отдельных параметров команды перемещения переместите курсор на нужный параметр и нажмите клавишу **F4 [CHOICE]** или введите числовое значение, подробнее на рисунках 23-25.

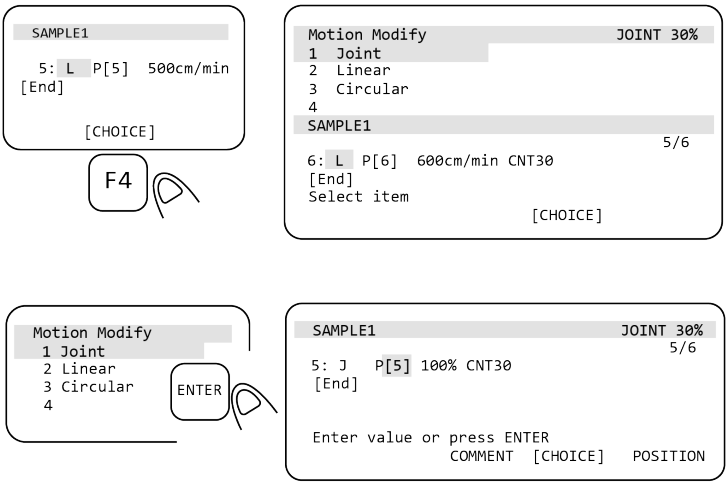

Рис. 23: Изменение типа перемещения

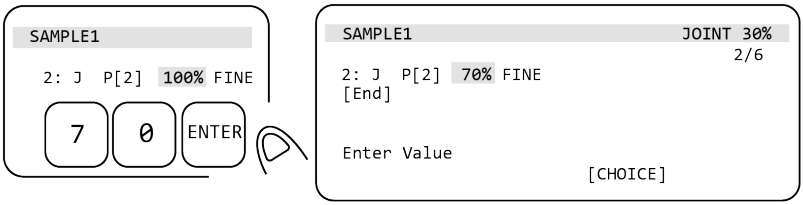

Рис. 24: Изменение скорости подачи

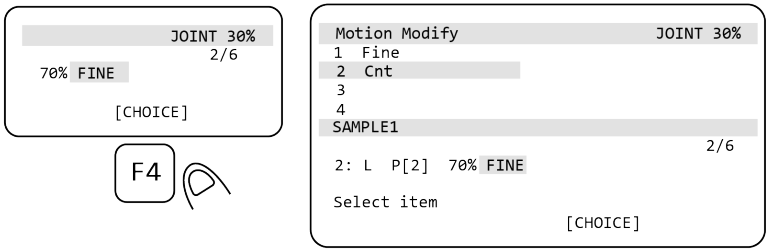

Рис. 25: Изменение типа позиционирования

4.3.7 Команда коррекции ускорения

Коррекция ускорения позволяет указать то, на сколько быстро будет ускорятся робот при перемещении по траектории. В команде перемещения обозначается как **ACC** и может принимать значения от **0** до **150**. Чем меньше значение **ACC**, тем меньше будет ускорение на траектории и тем более плавно будет перемещаться робот.

Формат команды

J P[1] 50% FINE ACC80

Для ввода команды коррекции ускорения установите курсор после команды перемещения и в появившемся меню справа выберете **ACC**.

4.4 Указание аргументов

После ввода команды вызова программы или функции может потребоваться указать аргументы. Подменю выбора аргументов открывается клавишей **F4 [CHOICE]**.

Примечание: количество и тип аргументов, передаваемых функции или программе, ни как не контролируется, в случае

неправильного указания аргументов программа завершится с ошибкой.

4.5 Команды перехода

Команды перехода служит для реализации циклов и ветвлений в программе.

4.5.1 Команда безусловного перехода

Команда метки (**LBL[i]**) используется для указания точки в программе, к которой должно перейти управление. Данная команда используется в паре с командой **JMP JBL[i]**.

Принцип работы таков: в коде программы ставится команда **LBL[i]**, вместо **i** пользователь присваивает номер по своему усмотрению (в пределах одной программы номер **i** должен быть уникален для каждой команды **LBL[i]**). В том месте кода, в котором должен быть совершен переход, пишется команда **JMP LBL[i]**. При этом вместо **i** ставиться номер той команды **LBL[i]**, к которой должен быть совершен переход. Во время исполнения кода, в тот момент, когда точка выполнения доходит до команды **JMP LBL[i]**, то она (точка выполнения) перемещается к команде **LBL[i]** с соответствующим номером.

Формат команды:

LBL [i: Comment] JMP LBL[i]

1. **i** - номер метки (**1**—**32786**);
2. **Comment** - комментарий (**1**—**16** символов);

4.5.2 Команда конца программы и подпрограммы

Команда **END** вставляется автоматически в конце программы и подпрограммы.

4.5.3 Команда CALL

CALL используется, что бы вызывать подпрограмму внутри программы. При вызове подпрограммы последней передаётся управление. После того как подпрограмма достигнет команды **[END]**, управление возвращается программе. Для того, что бы вставить команду **CALL**, нажмите **F1 [INS]**, в появившемся списке выберите **CALL**, затем выберите программу, которую необходимо вызвать.

Использование аргументов в команде CALL позволяет задать аргумент для вызываемой подпрограммы, тем самым осуществлять передачу данных между программой и подпрограммой. Пример использования аргументов можно увидеть на рисунке 26.

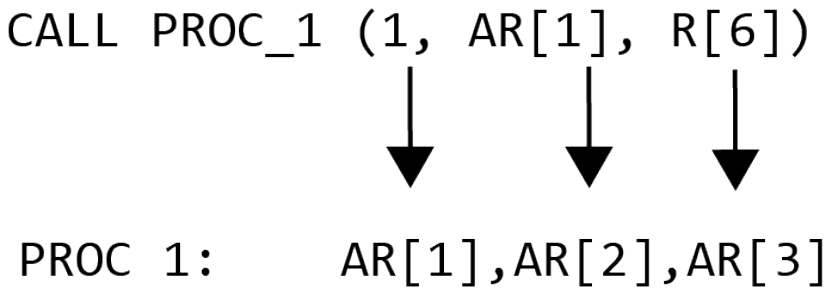

Рис. 26: Аргументы программы

4.5.4 Команды условного перехода

Команда условного перехода выполняет или не выполняет какие либо действия в зависимости от заданного условия.

Команды условного перехода бывают двух типов.

- Команда условного сравнения: вызывает передачу управления на указанную метку или в указанную программу при удовлетворении некоторого условия. Имеются команды условного сравнения регистра и условного сравнения ввода-вывода;

- Команда условного сравнения сигналов ввода вывода: вызывает передачу управления на заданную команду перехода или вызова подпрограммы в соответствии со значением регистра.

Для ввода в режиме обучения команды условного перехода нажмите клавишу **F1 [INST]**, чтобы вызвать подменю. Затем выберите в подменю элемент команды управления **IF/SELECT**.

Команда условного сравнения регистра **IF R[i] (оператор) (значение) (обработка)** сравнивает значение, хранящееся в регистре, с другим значением. Если условие сравнения удовлетворено, выполняет обработку (**JMP** или **CALL**).

Формат команды

IF R[i] (оператор) (значение) (обработка)

Оператор – символы:(>), (>=), (=), (<), (<=), (<>).

Значение – константа, регистр **R[i]**.

Обработка – **JMP LBL[i]**, **CALL** (подпрограмма).

Если содержимое регистра сравнивается с действительным значением при помощи оператора «=», оно не всегда может

соответствовать действительному значению из-за ошибок округления. При операциях с действительными (аналоговыми) величинами следует использовать другой символ, например, «<>».

Команда условного сравнения ввода-вывода периферийных устройств **IF (сигнал) (оператор) (значение) (обработка)** сравнивает значение сигнала ввода-вывода с другим значением. Если условие сравнения удовлетворено, выполняется заданная обработка.

Формат команды для обработки цифровых сигналов управления

IF (сигнал) (оператор) (значение) (обработка)

Сигнал – цифровой ввод **DI[i]**, цифровой вывод **DO[i]**.

Оператор – символы (=), (<>).

Значение – **ON**, **OFF**, **DI[i]**, **DO[i]**.

Обработка – **JMP LBL[i]**, **CALL**.

Формат команды для обработки аналоговых сигналов управления

IF (сигнал) (оператор) (значение) (обработка)

Сигнал – аналоговый ввод **AI[i]**, аналоговый вывод **AO[i]**.

Оператор – символы:(>), (>=), (=), (<), (<=), (<>).

Значение – константа, регистр **R[i]**.

Обработка – **JMP LBL[i]**, **CALL**.

Внимание! Для аналоговых сигналов знак «=» лучше не применять.

4.5.5 Команда множественного выбора

Данную команду удобно применять в тех случаях, когда необходимо обработать множество значений регистра.

Формат команды:

SELECT R[1] = (значение) (обработка)
= (значение) (обработка)
= (значение) (обработка)
ELSE (обработка)

4.6 Команды ожидания

Команда ожидания позволяет остановить выполнение программы на заданный промежуток времени или до выполнения заданного условия.

- Команда ожидания с ограничением по времени. Задерживает выполнение программы на заданный промежуток времени;
- Условная команда ожидания выполнения условия. Задерживает выполнение программы, пока не будет удовлетворено заданное условие или не истечет заданный период времени.

Для ввода команды ожидания в режиме обучения нажмите клавишу **F1 [INST]**, чтобы вызвать подменю. Затем выберите в подменю элемент команды управления **WAIT**.

4.6.1 Команда ожидания с ограничением по времени

Останавливает выполнение программы на заданное время, по истечению которого выполнение продолжается автоматически.

Формат команды

WAIT (время)

Время можно задать явно, например, **WAIT 10.5sec** или можно сослаться на регистр **R[i]**.

4.6.2 Команда ожидания выполнения условия

Команда ожидания выполнения условия **WAIT (условие) (обработка)** останавливает программ до выполнения условия. Обработка при этом не является обязательной частью команды, если её нет, то выполнение продолжится только после выполнения заданного условия

4.7 Редактирование программы

Редактирование программ осуществляется при помощи команд редактирования.

Список команд редактирования можно открыть в меню программы при помощи клавиш **NEXT [>]**, **F5 [EDCMD]**. В открывшемся списке можно выбрать нужную команду.

Insert (**вставка**). Вставляет указанное количество пустых строк в указанном месте.

Delete (**удалить**). Удаляет одну или несколько команд.

Copy (**копировать**). Копирует указанные строки.

Find (**найти**). Выполняет поиск в программе.

Replace (**заменить**). Заменяет один элемент программы другим.

Undo (**отменить**). Отменяет последнюю операцию, в том числе операцию отмены.

4.8 Система технического зрения iR Vision

Система технического зрения для робота iR Vision позволяет роботам «видеть». Данная системе может применяться, например, для определения местоположения деталей, позиции робота и так далее.

Принцип работы iR Vision состоит в следующем: на стороне **программы зрения** определяются расположения объектов и вычисляется их смещение относительно эталонного объекта, на стороне **программы движения** задаётся положение робота, соответствующее положению эталонного объекта (см. ниже).

Что бы перейти к настройке iR Vision, нажмите клавишу **MENU** → **iRVision** → **Vision Setup**. Появится меню настройки iR Vision, в котором есть два раздела:

1. **Camera Data**, в этом разделе отображается информация об установленных в системе камерах. В нашем случае это единственная **CELL_CAM**, выбрав её клавишей **ENTER** можно перейти к настройкам камеры;

2. **Visual Process Tools**, тут отображаются программы, интерпретирующие изображение с камеры. С помощью клавиш **F2 [CREATE]** и **F3 [EDIT]** можно создать новую программу или редактировать уже существующую соответственно.

4.8.1 Создание программы распознавания

Для создания программы зрения необходимо перейти в соответствующее меню (**F2 [CREATE]**), в котором пользователю предлагается выбрать камеру и тип программы. Так как мы имеем всего одну, жёстко закреплённую камеру, то всегда выбираем пункт **2-D Single-View Vision Process**.

На следующем шаге выбираем называем программу (обязательно) и даём комментарий (опционально). После того как программа названа мы автоматически перейдём в меню **Vision Setup**. Теперь можно редактировать свеж созданную программу, с помощью клавиши **F3 [EDIT]**.

Далее настройте пункты с **Tree** (см. ниже) в соответствии с задачей вашей программы. Ниже представлено описание пунктов меню редактирования.

В меню редактирования экран разделён на две части:

1. С лева в верхней части находятся две вкладки:
   (a) **Image** – здесь отображается изображение с камеры;
   (b) **Tree** – иерархический список всех компонентов программы (рисунок 27).

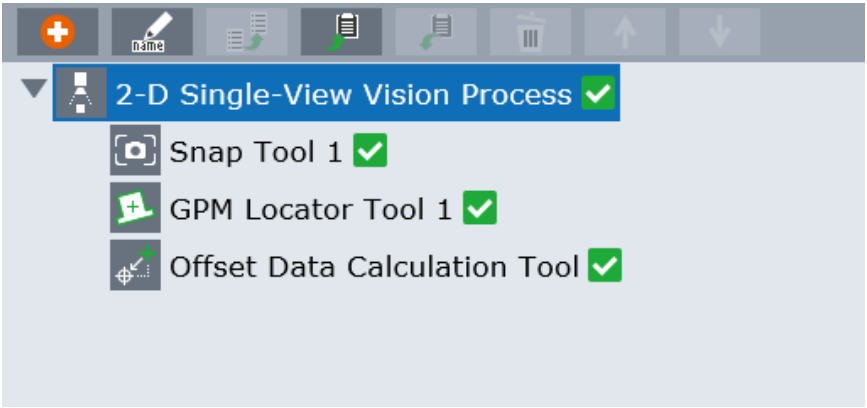

Рис. 27: Список компонентов программы зрения, на рисунке представлены **не** все возможные компоненты.

2. В правой части располагается выделенный в списке **Tree** пункт;
3. Клавиши в нижней части экрана отвечают за:
   (a) **F2 [LIVE]** – включает/отключает изображение с камеры в реальном времени. При включённом режиме

LIVE невозможно изменять настройки камеры и распознавать объекты!;

(b) **F3 [SNAP]** – делает мгновенный снимок, с которого в дальнейшем распознаются объекты;

(c) **F4 [FIND]** – отдаёт команду на поиск объектов на текущем снимке (см. пункт 3b);

(d) **F5 [END EDIT]** – клавиша выхода из режима редактирования программы.

Пройдёмся по основным пунктам в разделе **Tree**:

1. **2-D Single-View Vision Process** – содержит основные настройки программы распознавания:

   (a) **Camera** – выбор камеры, с которой будет работать программа;

   (b) **Number to Find** – количество объектов, которые будет искать программа;

   (c) **Offset Mode** – задаёт режим распознавания и вычисления координат объекта. Существуют два основных режима:

      i. **Fixed Frame Offset** – вычисляет координаты объекта, расположенного неподвижно;

      ii. **Tool Offset** – вычисляет координаты объекта в **СК TOOL**.

   (d) **Offset Frame** – задаёт **СК**, в которой вычисляется положение объекта в режиме **Fixed Frame Offset**.

2. **Snap Tool** – этот пункт отвечает за настройки камеры. Здесь задаётся то, как будет обрабатываться изображение с камеры перед тем, как начнётся распознавание объектов;

(a) **Snap Window** – позволяет обрезать “лишнюю” часть изображения с камеры. Кнопка **Set** позволяет начать кадрирование;

(b) **Resolution Reduction** – кратно уменьшает выходное разрешение изображения, это необходимо для некоторых режимов распознавания;

(c) **Image Size** – отображает текущее разрешение;

(d) **Exposure Mode** – выбор режима экспозиции существуют три варианта:

i. **Fixed** – время экспозиции фиксировано и задаётся в пункте Exposure Time;

ii. **Auto** – время экспозиции выбирается автоматически;

iii. **HDR** – режим высокого динамического диапазона.

3. **GPM Locator Tool** – настраивает распознавание:

(a) Кнопка **Teach** - позволяет задать объект, которые будет искать программа распознавания. Для задания объекта его необходимо обвести в рамку;

(b) **Model ID** - используется для распознавания нескольких различных фигур, для каждой фигуры указываться свой **Model ID**;

(c) **Score Threshold** - этот параметр определяет то, на сколько найденный объект должен быть "похож" на заданный, что бы попасть в конечный результат распознавания. Изменяется от **0** до **100**, чем меньше значение, тем менее "похожим" может быть объект. Уменьшайте этот параметр, если не все объекты в поле зрения камера распознаются, увеличивайте его, если посторонние детали распознаются в качестве объектов;

(d) **Contrast Threshold** - определяет уровень контраста, необходимый для определения объекта. Если изображение объекта недостаточно контрастное (тёмный объект на черном фоне), то уменьшите этот параметр. Увеличение этого параметра поможет избежать распознавания посторонних малоконтрастных объектов;

(e) **Area Overlap** - позволяет избегать перекрытий двух объектов на изображении. Чем этот параметр больше, тем более "сильное" перекрытие объектов допускает программа распознавания;

(f) **Search Window** - задаёт область поиска объектов. По умолчанию областью поиска является всё изображение целиком, с помощью кнопки **Set** можно задать другую область изображения, обведя её в рамочку.

4. **Offset Data Calculation Tool** – эта часть отвечает непосредственно за вычисление координат найденных деталей:

   (a) **Ref. Data To Use** – имеет два положения:

      i. **Static** – используется, если все объекты одинаковые;

      ii. **Model ID** – для различающихся объектов.

   (b) **Part Z Height** – высота объекта;

   (c) **Ref. Pos. Status** – положение эталонного объекта. Для того чтобы выбрать эталонный объект, необходимо:

      i. Выполнить поиск объектов клавишами **F3 [SNAP]**, затем **F4 [FIND]**;

      ii. В верхней левой часть переключиться на вкладку **Image**;

iii. В нижнем левом углу развернуть вкладку **Results**;

iv. В открывшейся секции выбрать объект из списка, выбранный объект будет выделен на изображении;

v. После того, как объект выбран, нажать кнопку **Set**.

4.8.2 Использование программы распознавания

Для того чтобы воспользоваться iR Vision, непосредственно при написании программы необходимо:

1. Установить в рабочее пространство эталонный объект, не двигать его до конца настройки iR Vision;

2. Задать «ждущее» положение робота **P[1]**, в котором робот не будет попадать в поле зрения камеры, в качестве такового подойдёт домашнее положение;

3. Задать положение **P[2]**, ровно над эталонным объектом, на высоте 10 см;

4. Задать положение **P[3]**, в котором робот установлен в позицию для захвата эталонного объекта;

5. После того, как положения **P[1]**, **P[2]** и **P[3]** определенны в программе, необходимо перевести робота в положение **P[1]**.

6. Создать программу распознавания (раздел 4.8.1);

7. Получить данные позиционирования из программы распознавания в основной программе:

   J P[1] 100% FINE; //перевод робота в ждущее положение

   VISION RUN_FIND 'WORK1'; //вызов программы зрения **'WORK1'**

VISION GET_OFFSET ‘WORK1’ VR[1] JMP LBL[99]; //получение данных смещения из **‘WORK1’** и запись в регистр **VR[1]** (можно использовать любой другой свободный регистр), если программе зрения не удалось найти объекты, то выполниться команда **JMP LBL[99]** (можно использовать и другой номер)

J P[2] 100% CNT VOFFSET, VR[1]; //перемещает захват робота в точку над одним из объектов

L P[3] 100mm/sec FINE VOFFSET, VR[1]; //перемещает робота в позицию для захвата

CALL HAND_CLOSE; //сжимает захват, помимо сжатия захвата тут могут быть любые действия с объектом

L P[2] 100mm/sec FINE VOFFSET, VR[1]; //поднимает захват с деталью

5 КОМАНДЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Существуют два режима выполнения.

Пошаговой режим работы – это построчное выполнение программы. После того как будет выполнена одна строка программы, следующая строка выполниться только после нажатия клавиши **FWD**.

Непрерывный режим работы – это выполнение программы без остановок. После выполнения одной строки, следующая начнёт выполняться незамедлительно. В режимах **T1** и **T2** пользователь может остановить исполнение программы отпустив клавишу **SHIFT** или **переключатель аварийной блокировки**. В режимах **T1**, **T2** и **AUTO** можно остановить программу **кнопкой аварийной блокировки**.

Индикатор режима выполнения программы **STEP** расположен в верхней части экрана пульта обучения. Зелёный цвет индикатора означает непрерывный режим работы. Желтому цвету соответствует пошаговый режим. Запустить программу можно двумя способами: с помощью пульта обучения (клавиши **SHIFT** + **FWD**) или клавишей **START** с панели оператора.

5.1 Запуск программы с пульта обучения

1. Установите **трёхпозиционный переключатель** на пульте оператора в положение **T1** или **T2** (рисунок 5);

2. Включите пульт обучения;

3. Установите скорость перемещения рабочих органов (рисунок 7);

4. Нажмите клавишу **SELECT (выбор)**, в открывшемся списке выберите программу, затем нажмите клавишу **ENTER**;

5. Клавишей **STEP** выберите пошаговый или непрерывный режим работы;

6. Переместите курсор в строку с которой хотите начать выполнение;

7. Удерживая **переключатель аварийной блокировки** нажмите клавишу **RESET**, индикатор **FAULT** погаснет. Не отпускайте **переключатель аварийной блокировки** до конца выполнения программы;

8. Нажмите и удерживайте клавишу **SHIFT**, затем **FWD**. Начнётся исполнение программы.

5.2 Запуск программы с панели оператора

1. Установите трёхпозиционный переключатель на пульте оператора в положение **AUTO** (рисунок 5);

2. Задайте нужную скорость перемещения рабочих органов (рисунок 7);

3. Нажмите клавишу **SELECT (выбор)**, в открывшемся списке выберите программу, затем нажмите клавишу **ENTER**;

4. Клавишей **STEP** установите пошаговый или непрерывный режим работы;

5. Переместите курсор в первую строку программы;

6. Пульт обучения должен быть выключен;

7. На панели оператора нажмите **CYCLE START** (рисунок 3).

6 ПРИМЕРЫ ПРОГРАММ

В данном разделе будут рассмотрены несколько примеров написания программ с подробными пояснениями.

6.1 Построение башни

Напишем простейшую программу, которая позволит строить башню из цилиндров. Для этого выполним следующие действия:

0. Расставить цилиндры в исходные позиции;

1. Включить робота согласно инструкции (раздел 2.4), в режиме отладки **T1**;

2. Зарегистрируйте новую программу (раздел 4.1).

3. Для того, чтобы перед выполнением программы робот вернулся в домашнее положение необходимо в самом начале вызвать программу **AA_HOME** с помощью оператора **CALL**. Это позволит роботу каждый раз начинать выполнение программы с одного и того же положения;

4. Аналогично перед началом работы необходимо подготовить захват, для этого вызываем макрос **HAND_OPEN** с помощью оператора **CALL**, в результате мы получим основу для будущей программы;

5. Для дальнейшего удобства рекомендуется работать в системе координат **USER 1**, которая совпадает с мировой системой координат **WORLD**, дальнейшее изложение приведено именно для СК **USER 1**;

6. Для того, чтобы робот взял первый цилиндр необходимо поставить захват робота над цилиндром, после чего добавьте

точку перемещения в режиме **L[] CNT100**, нажав на экране **ADD MOVE POINT**;

7. Теперь можно опустить захват на цилиндр, и добавить ещё одну точку перемещения в режиме **L[] FINE**. Теперь мы готовы к тому, чтобы активировать захват, для этого оператором **CALL** вызовем макрос **HAND_CLOSE** и выполним соответствующую строку, с помощью сочетания клавиш **SHIFT** + **FWD**, в результате робот возьмёт цилиндр;

8. Поднимем захват робота вместе с цилиндром и добавим ещё одну точку в режиме **L[] CNT100** (рекомендуется поднимать цилиндр так, чтобы его нижняя часть была выше верхних частей остальных цилиндров, для предотвращения возможных столкновений);

9. Переместим захват над тем местом, где должна появиться будущая башня, добавим точку **L[] CNT100**, после чего опустим захват до касания нижней части цилиндра, добавим точку **L[] FINE** (рекомендуется запомнить/записать x и y координаты этой точки, чтобы в дальнейшем опускать цилиндры ровно, по координатам). Теперь мы готовы поставить цилиндр на место, для этого добавим оператор **WAIT .20 (sec)**, это необходимо, чтобы робот успел полностью остановиться, прежде чем отпустить цилиндр;

10. Вызовем макрос **HAND_OPEN**, чтобы разжать захват, тем самым отпустив цилиндр. Поднимем захват над первым этажом башни и добавим точку **L[] CNT100**. Теперь наша программа выглядит примерно так:

```
CALL AA_HOME
CALL HAND_OPEN

L P[1] 100mm/sec CNT100
L P[2] 100mm/sec FINE
CALL HAND_CLOSE
L P[3] 100mm/sec CNT100
L P[4] 100mm/sec CNT100
L P[5] 100mm/sec FINE
WAIT .20(sec)
CALL HAND_OPEN
L P[6] 100mm/sec CNT100
[End]
```

11. Аналогично повторите пункты 5 – 9 для остальных цилиндров.

6.2 Построение башни с применением iR Vision

1. Повторите пункты 0-2 из предыдущей программы;

2. Создадим новую программу распознавания, в качестве эталонного укажите один из цилиндров и **не перемещайте его** в процессе написания программы (раздел 4.8.1);

3. Добавим команду выбора системы координат **UFRANE_NUM=9**, в качестве системы координат выберем ту, которую использовали в программе распознавания, в нашем примере это СК номер 9;

4. Для дальнейшего удобства будем использовать регистры положения. Нам понадобятся **три регистра положения**

и один **обычный регистр**. Для начала поместим в захват робота один из цилиндров, и переместим захват вместе с цилиндром в то место, где должна стоять башня, после этого можно записать текущую позицию в один из свободных регистров положения (так как номера регистров можно брать любые, то далее будем оперировать только названиями регистров), назовём его **bottom of the tower**;

5. Из предыдущего положения поднимем робота вверх, выше, чем полная башня из всех цилиндров. Запишем это положение в новый регистр положения, назовём его **top of the tower**;

6. Третий регистр положения назовём **current tower height**;

7. Теперь возьмём обычный регистр, назовём его **height_of_cylindr** и присвоим ему значение 40, это высота цилиндра в миллиметрах (расстояния для робота указывается в миллиметрах);

8. Сразу после строки с выбором системы координат пропишем **PR[52:current tower height]=PR[51:bottom of the tower]**, эта команда скопирует значение регистра положения **bottom of the tower** в регистр **current tower height**;

9. Далее начнём цикл с команды **LBL[1]**;

10. В цикле вызовем программы **AA_HOME** и **HAND_OPEN**, с помощью оператора **CALL**;

11. Вызовем программу распознавания и получим из неё данные командами:
VISION RUN_FIND
'FIND_CYL_IN_HOLES' и

VISION GET_OFFSET
'FIND_CYL_IN_HOLES' VR[1] JMP LBL[99]. В нашем случае используется программа распознавания **'FIND_CYL_IN_HOLES'**, а данные записываются в регистр **VR[1]**. Если программа распознавания не найдёт ни одного объекта, то будет выполнена команда **JMP LBL[99]**, это будет условие выхода из цикла;

12. С помощью ручного управления поместим захват робота точно над эталонным объектом, на высоте 10см, добавим в программе команду линейного перемещения в эту точку **L P[1] 100mm/sec FINE**, для того чтобы применить к этой команде смещение, полученное из программы распознавания, добавьте в конце команды **VOFFSET, VR[1]**;

13. Опустите захват на цилиндр и, аналогично предыдущему пункту, добавьте точку движения со смещением;

14. Вызовете макрос сжатия захвата:
CALL HAND_CLOSE;

15. Скопируйте команду движения из пункта 12, в последних четырёх пунктах робот передвигается к цилиндру, берёт его в захват и приподнимает;

16. Теперь, когда робот захватил цилиндр, переместим его в домашнее положение (**CALL AA_HOME**);

17. Добавим команду перемещения в регистр положения: **L PR[50:top of the tower] 100mm/sec CNT100**;

18. Потом в следующий: **L PR[52:current tower height] 100mm/sec FINE**;

19. Теперь можно раскрыть захват и поставить цилиндр: **CALL HAND_OPEN**;

20. Далее обновим координаты текущей высоты башни.

    PR[52, 3:current tower height]=
    PR[52, 3:current tower height]+
    R[50:height_of_cylindr]

21. Добавим команду перемещения **L PR[52:current tower height] 100mm/sec CNT100**;

22. Замкнём цикл, добавив переход к началу: **JMP LBL[1]**;

23. Добавим флаг для выхода из цикла: **LBL[99]**;

24. Конец **[End]**.

Итоговый код программы:

```
UFRAME_NUM=9
PR[52:current tower height]=
PR[51:bottom of the tower]

LBL[1]

CALL AA_HOME
CALL HAND_OPEN

VISION RUN_FIND
'FIND_CYL_IN_HOLES'
VISION GET_OFFSET
'FIND_CYL_IN_HOLES' VR[1]
```

```
JMP LBL[99]

L P[1] 100mm/sec CNT100
VOFFSET, VR[1]
L P[2] 100mm/sec FINE
VOFFSET, VR[1]
CALL HAND_CLOSE
L P[1] 100mm/sec CNT100
VOFFSET, VR[1]

CALL AA_HOME

L PR[50:top of the tower]
100mm/sec CNT100
L PR[52:current tower height]
100mm/sec FINE
CALL HAND_OPEN
PR[52, 3:current tower height]=
PR[52, 3:current tower height]+
R[50:height\_of\_cylindr]
L PR[52:current tower height]
100mm/sec CNT100

JMP LBL[1]

LBL[99]

[End]
```

7 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислить основные узлы и агрегаты робота, назвать их назначение.

2. Рассказать порядок включения и выключения робота.

3. Какие существуют режимы работы контроллера? Для каких задачи используется каждый из режимов?

4. Как изменять скорость движения робота в ручном режиме? Какие скорости бывают?

5. Назвать системы координат, используемые роботом, объяснить их суть.

6. Что такое **TCP**?

7. Перечислить последовательность действий для ручного перемещения робота.

8. Что такой регистры и для чего они используются? Какие бывают виды регистров? Какие операции можно выполнять с регистрами?

9. Какой формат имеет команды перемещения? Какие существуют параметры команд перемещения и за что они отвечают?

10. Что такое iR Vision? Для чего используется?

11. Какие есть команды редактирования программы?

12. Перечислить порядок действий для запуска программы в различных режимах работы контроллера.

8 ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

1. Написать программу, которая строит две башни из цилиндров, а затем разбирает их.

2. Поднять цилиндр, перевернуть его на 90°вокруг горизонтальной оси, и поставить на место

   (a) в ручном режиме.

   (b) написать отдельную программу.

3. Построить башню из двух или трёх цилиндров и переместить её целиком.

   (a) в ручном режиме.

   (b) написать отдельную программу.

4. Зажать в захват карандаш и задать такую систему координат инструмента, начало которой совпадает с концом карандаша, а ось y параллельна с карандашом.

5. Положить в камеру робота лит бумаги и задать пользовательскую систему координат, оси которой были бы параллельны сторонам листа.

6. Написать программу, которая двигает захват по окружности.

7. Зажать карандаш в захват и написать программу, которая рисует заданную фигуру на листе бумаги.

   (a) Окружность.

   (b) Квадрат.

   (c) Звезда.

8. Составить программу распознавания различных объектов для iR Vision:
   (a) Лист бумаги.
   (b) Линейка.
   (c) Цилиндр, лежащий на боку.
9. Составить программу с использованием iR Vision, которая перемещает цилиндры из одной группы лунок в другую.

9 СПИСОК ЛИТЕРАТУРЫ

- Балабанов, П. В. Программирование робототехнических систем: учебное электронное издание : учебное пособие / П. В. Балабанов. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2018. – 82 с.

- Программирование для системы ЧПУ Fanuc Oi: учебное пособие / А.С. Александров, Д.В. Васильков, В.В. Голикова; Балт. гос. техн. ун-т – СПб., 2019. – 142 с.

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

Учебное издание

Составитель:
Шкляев Константин Янович

Программирование в образовательном робототехническом комплексе FANUC

Учебно-методическое пособие

Авторская редакция

Подписано в печать 12.12.2023. Формат 60x84 1/16.
Усл. печ. л. 4,2. Уч. изд. л. 2,9.
Тираж 28 экз. Заказ № 2177.

Издательский центр «Удмуртский университет»
426034, г. Ижевск, ул. Ломоносова, 4Б, каб. 021
Тел. : + 7 (3412) 916-364, E-mail: editorial@udsu.ru

Типография Издательского центра «Удмуртский университет»
426034, г. Ижевск, ул. Университетская, 1, корп. 2.
Тел. 68-57-18