Санкт-Петербургский политехнический университет Петра Великого

Научно-исследовательская работа

на тему:

«Управление промышленным манипулятором KUKA с механическим захватом»

Выполнил: Воробьев Д. М.

Руководитель: Ананьевский М.С.

Группа: 3331506/10101

Санкт-Петербург

2024

Оглавление

Теоретические сведения

Структура программы

Программа на языке KRL состоит из двух файлов: файла кода (.src) и файла данных (.dat) с одинаковым именем. Файл кода содержит код программы, а файл данных — описания переменных, точек, массивов и т. д. для оптимизации размера файла кода.

Файл кода состоит из главной и дополнительных функций. Название главной функции должно совпадать с именем файла кода.

Функция на языке KRL разделена на три раздела:

1. Объявление (объявление переменных, массивов, точек, структур и т. д.).
2. Инициализация (инициализация переменных, массивов и т. д., объявленных в разделе объявления).
3. Инструкции (основной код программы).

Объявление функции начинается с ключевого слова DEF и заканчивается словом END.

Файлы программы на языке KRL называются с использованием одного и того же имени, например, “TEST.SRC” и “TEST.DAT”, где имя файла (в данном случае “TEST”) совпадает с названием главной функции в файле кода.

Круглые скобки после имени функции сообщают интерпретатору KRL, что ваша программа использует функцию.

Переменные и типы данных

В языке KRL используются простые и структурированные типы данных. Для объявления переменных рекомендуется использовать ключевое слово “DECL”, которое выделяет память для переменной и предотвращает возможные ошибки. Хотя использование “DECL” не всегда обязательно, оно повышает надежность кода. Простые типы данных в KRL (таблица 1.2.1.1) включают в себя основные типы, встречающиеся в большинстве языков программирования. Объявление переменных в KRL похоже на объявление в других языках, за исключением использования ключевого слова “DECL”.

*Таблица **Ошибка! Текст указанного стиля в документе отсутствует.**.1 - Простые типы данных*

Типы данных	Целое	С точкой	Логический	Символьный
Ключевое слово	INT	REAL	BOOL	CHAR
Диапазон значений	$-2^{31} \dots 2^{31} - 1$	$\pm 1.1E - 38 \dots$ $\pm\ 3.4E + 38$	TRUE, FALSE	ASCII character

Объявим несколько переменных.VAR_1 и VAR_2:

```
DEF TEST()
;------------Объявление---------
DECL INT VAR_1
DECL REAL VAR_2
;------------------------------
;------------Инициализация------
;------------Основной код-------
END
```

В этом примере переменная VAR_1 имеет тип INT (целое число), а VAR_2 - тип REAL (вещественное число). KRL также поддерживает массивы данных.

Для объединения различных типов данных в языке KRL используется оператор STRUC, который позволяет создавать новые составные типы данных. Типичным примером составного типа данных является POS,

объявленный в файле $OPERATE.SRC. Он состоит из шести переменных типа REAL и двух типа INT. Например, чтобы объявить переменную POSITION типа POS, нужно использовать следующую запись: DECL POS POSITION. Для присвоения значений отдельным элементам структуры можно использовать оператор точки, который позволяет обращаться к каждому элементу структуры отдельно:

POSITION.X = 28

POSITION.Y = 10

POSITION.Z = 5

POSITION.A = 0

POSITION.B = 90

POSITION.C = 0

или совместно при помощи «совокупности»:

POSITION = {X 28, Y 10, Z 5, A 0, B 90, C 0}

Эти две записи выполняют одинаковую инициализацию.

Также тип данных может указываться в начале инициализации структуры.

Например:

```
POSITION={POS: X 230,Y 0.0,Z 342.5}
```

В языке KRL уже заранее определены основные структурные типы – табл. 1.2.2.1

*Таблица **Ошибка! Текст указанного стиля в документе отсутствует.**.2 - Типы структур*

```
STRUC AXIS    REAL A1,A2,A3,A4,A5,A6
STRUC E6AXIS  REAL A1,A2,A3,A4,A5,A6,E1,E2,E3,E4,E5,E6
STRUC FRAME   REAL X,Y,Z,A,B,C
STRUC POS     REAL X,Y,Z,A,B,C, INT S,T
STRUC E6POS   REAL X,Y,Z,A,B,C,E1,E2,E3,E4,E5,E6, INT S,T
```

Программирование в среде WorkVisual

Подключение SmartPad

Сначала необходимо соединить компьютер с контроллером KUKA через порт KLI с помощью Ethernet-кабеля. Затем нужно настроить IP-адрес компьютера так, чтобы он соответствовал IP-адресу робота. Для этого необходимо нажать на значок робота на SmartPad, затем зайти в раздел Start-up и выбрать Network Configuration. IP-адрес робота – 172.31.1.147, поэтому мы установили адрес 172.31.1.148 для компьютера, чтобы они были в одном диапазоне, но не совпадали.

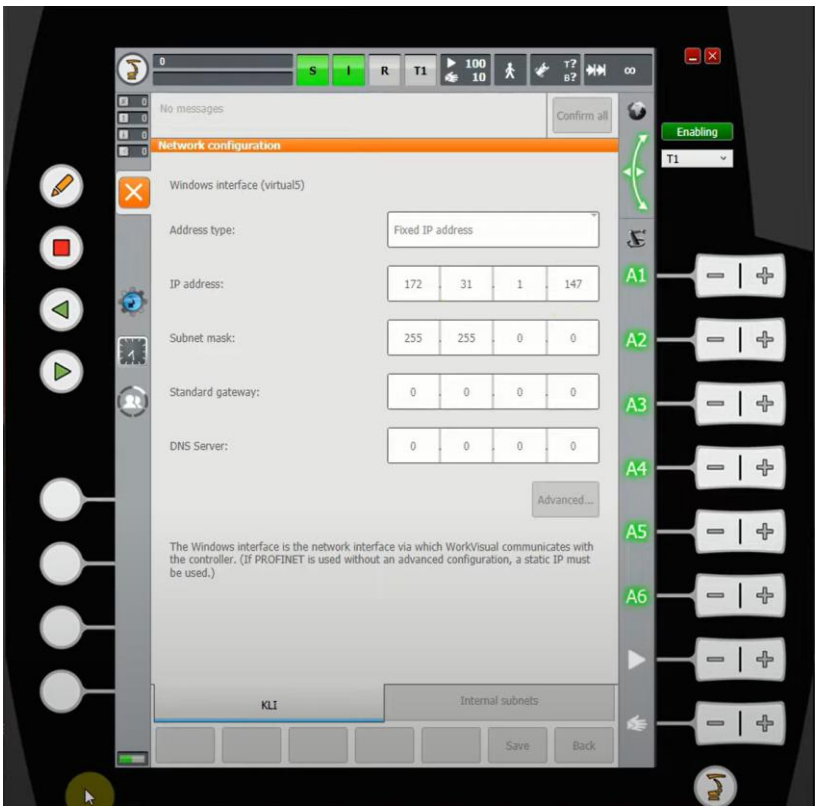

Рисунок 2.1.1 – IP-адрес робота

Для правильной работы с контроллером KUKA необходимо убедиться, что язык системы совпадает с языком SmartPad. В нашем случае это английский.

WorkVisual - это программа, которая позволяет скачивать, редактировать, удалять и создавать новые проекты для робота, а затем загружать их обратно в память робота. Чтобы иметь возможность принимать проекты, выгруженные из WorkVisual, необходимо установить на SmartPad уровень доступа Expert. Это можно сделать в разделе Users, используя стандартный пароль KUKA.

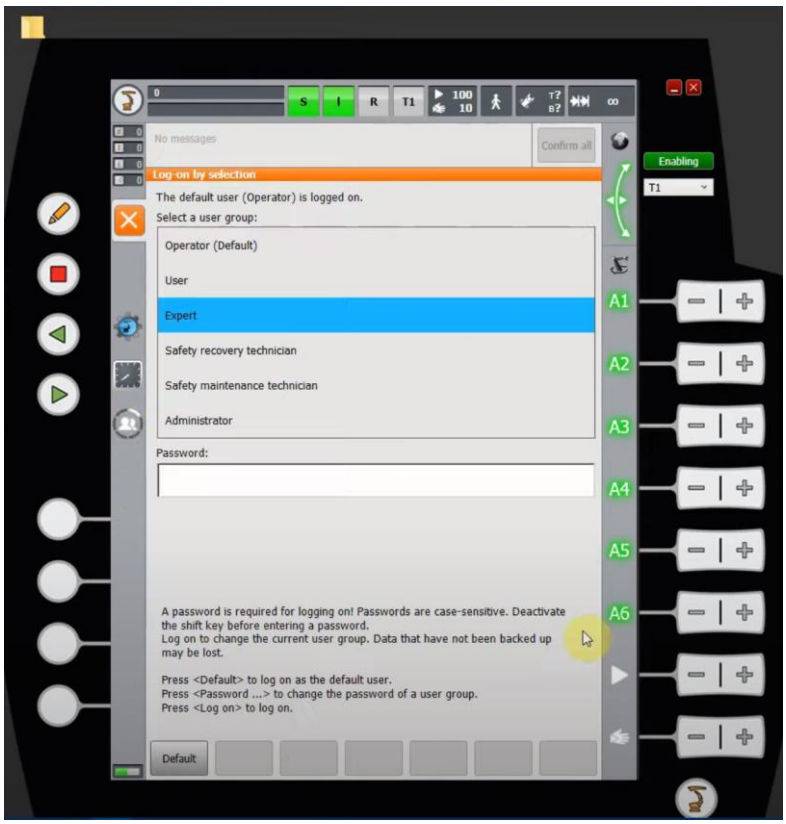

*Рисунок **Ошибка! Текст указанного стиля в документе отсутствует.**.1.2 - Установка режима Expert*

Режимы выполнения программы

Робот обладает тремя режимами работы, выбираемыми через диспетчер переключений на SmartPad:

- **Режим T1** (ручной, сниженная скорость): Предназначен для тестирования, программирования и обучения. Обеспечивает пониженные скорости движения.
- **Режим T2** (ручной, высокая скорость): используется для тестирования, обеспечивает скорость, соответствующую запрограммированной.
- **Режим AUT** (автоматический): применяется для нормальной эксплуатации.

В процессе работы мы применяли режим T1 для тестирования программы и режим AUT для демонстрации ее работы в реальных условиях.

Определение координат манипулятора

Чтобы установить базу для манипулятора, необходимо определить его текущие координаты. Для этого:

1. На SmartPad перейдите в меню: Config -> Display -> Actual position.
2. На экране отобразятся координаты манипулятора относительно центра его крепления к столу.
3. Эти координаты будут обновляться в реальном времени, отображая как ход программы, так и ручное перемещение манипулятора.

Используя эти данные, можно определить подходящую точку для установки базы.

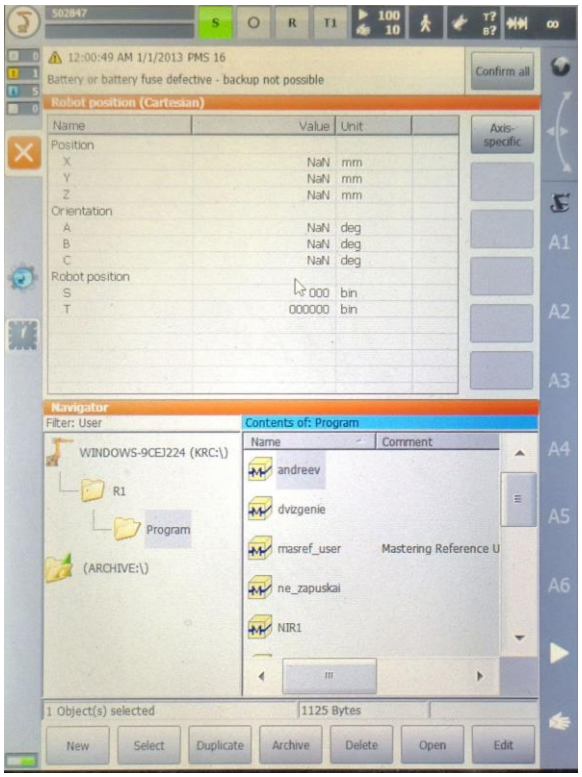

*Рисунок **Ошибка! Текст указанного стиля в документе отсутствует.**.3 – Определение позиции захвата*

Программа построения столба

Весь код программы представлен в приложении к курсовой работе, здесь же отмечаются основные элементы данной программы

```
DEF  tower ( )
DECL FRAME HOME_1
DECL FRAME START_TOWER
DECL FRAME START_KUBE
```

```
DECL FRAME MOVE_DOWN
DECL INT KUBE
DECL INT COUNTER
```

Этот блок кода, называемый “Блок объявления”, служит для объявления функций, переменных и структур данных, которые будут использоваться в программе.

Важно, что все объявления переменных должны быть сделаны до блока инициализации. Это означает, что перед тем, как присвоить переменным значения, их нужно сначала объявить.

Структура Frame - это предопределенный тип данных, который используется для хранения координат в пространстве. Он содержит шесть координат: X, Y, Z, OZ, OY, OX.

Иными словами, этот блок задает основы для работы программы, определяя все необходимые компоненты, которые будут использоваться в последующем коде.

```
;FOLD INI
  ...
;ENDFOLD (INI)
```

Далее идет блок инициализации о котором было сказано в 1 пункте.

```
OPEN_GRAB()
KUBE = 30
$BASE = $WORLD

HOME_1 = {X 94.9, Y 862.6, Z 43.9, A 146, B 0, C 0}
START_TOWER = {X 0, Y 300, Z 0, A 0, B 0, C 0}
START_KUBE = {X 0, Y 0, Z 0, A 0, B 0, C 0}

$BASE = HOME_1
```

В этом разделе программы мы задаем ключевые параметры для работы:

1. Размер кубика: определяем размеры кубика, который будет использоваться в задаче.
2. Базовая точка: указываем, от какой точки на рабочем пространстве будет вестись работа.
3. Координаты начала столба: указываем точку, где будет располагаться столб из кубиков.

4. Координаты начала линии кубиков: Указываем точку, где будет начинаться линия, по которой будут располагаться кубики.

Таким образом, мы устанавливаем начальные условия для размещения кубиков в заданном пространстве.

```
FOR COUNTER = 0 TO 5 STEP 1
START_KUBE.X = START_KUBE.X - KUBE - 8

MOVE_DOWN = START_KUBE
MOVE_DOWN.Z = MOVE_DOWN.Z + 50

PTP MOVE_DOWN
PTP START_KUBE
CLOSE_GRAB()
PTP MOVE_DOWN

MOVE_DOWN = START_TOWER
MOVE_DOWN.Z = MOVE_DOWN.Z + 50

PTP MOVE_DOWN
PTP START_TOWER
OPEN_GRAB()
PTP MOVE_DOWN
START_TOWER.Z = START_TOWER.Z + KUBE
ENDFOR
```

Этот код представляет собой цикл, который строит столб из кубиков.

Основные принципы:

1. Горизонтальная сборка: Кубики добавляются один за другим по горизонтали, формируя стенку.
2. Безопасный заход: Для того, чтобы не сбить уже установленные кубики, манипулятор заходит на них сверху (точка MOVE_DOWN).
3. Масштабируемость: Чтобы построить столб из большего количества кубиков, просто нужно увеличить значение в цикле FOR.

Разборка столба:

Обратный процесс: Разборка столба реализуется аналогично сборке, но в обратном порядке - кубики снимаются один за другим.

Таким образом, этот код обеспечивает удобный и безопасный механизм для сборки и разборки столба из кубиков.

```
DEF CLOSE_GRAB ()
$OUT[3]=TRUE
WAIT SEC 1.5
$OUT[3]=FALSE
END

DEF OPEN_GRAB ()
$OUT[4]=TRUE
WAIT SEC 1.5
$OUT[4]=FALSE
END
```

Управление захватом манипулятора осуществляется с помощью сигналов OUT[3] и OUT[4].

Закрытие захвата: Подача логической единицы (1) на OUT[3] приводит к закрытию захвата. Важно отметить, что длительное удержание сигнала (более 2 секунд) может привести к неработоспособности захвата, требуя перезагрузки всего робота. Оптимальное время для полного закрытия захвата - 1,5 секунды.

Открытие захвата: Подача логической единицы (1) на OUT[4] открывает захват.

Таким образом, для надежной работы захвата необходимо использовать сигналы OUT[3] и OUT[4] с заданным временным интервалом, чтобы избежать его поломки.

Заключение

В ходе работы была проведена настройка программной среды Work Visual. Была установлена и настроена программа, скорректирована база робота для точного позиционирования и написана программа для сборки столба из кубиков с использованием захвата. Программа была успешно протестирована на малых и больших скоростях, подтвердив точность движений, стабильную работу захвата и высокую производительность.

Список использованных источников

1. https://wikis.utexas.edu/display/SOAdigitech/KUKA+Programming+KRL+Examples
2. https://drstienecker.com/tech-332/11-the-kuka-robot-programming-language/
3. https://swsu.ru/sveden/files/PROGRAMMIROVANIE_PROMYSHLENNOGO_ROBOTA_KUKA_LAB.pdf
4. https://www.youtube.com/watch?v=GtxShP_Wtec&t=171s&ab_channel=FutureRobotics

Приложение

```
DEF  tower ( )
DECL FRAME HOME_1
DECL FRAME START_TOWER
DECL FRAME START_KUBE
DECL FRAME MOVE_UP
DECL FRAME MOVE_DOWN
DECL INT KUBE
DECL INT COUNTER

;FOLD INI
  ;FOLD BASISTECH INI
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI
    ;Make your modifications here

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP,
2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
;ENDFOLD

  OPEN_GRAB()
        KUBE = 30
        $BASE = $WORLD

        HOME_1 = {X 94.9, Y 862.6, Z 43.9, A 146, B 0, C 0}
        START_TOWER = {X 0, Y 300, Z 0, A 0, B 0, C 0}
        START_KUBE = {X 0, Y 0, Z 0, A 0, B 0, C 0}

        $BASE = HOME_1

        FOR COUNTER = 0 TO 5 STEP 1
          START_KUBE.X = START_KUBE.X - KUBE - 8

          MOVE_DOWN = START_KUBE
          MOVE_DOWN.Z = MOVE_DOWN.Z + 50

          PTP MOVE_DOWN
                PTP START_KUBE
                CLOSE_GRAB()
                PTP MOVE_DOWN

                MOVE_DOWN = START_TOWER
                MOVE_DOWN.Z = MOVE_DOWN.Z + 50
```

```
          PTP MOVE_DOWN
                PTP START_TOWER
                OPEN_GRAB()
                PTP MOVE_DOWN
                START_TOWER.Z = START_TOWER.Z + KUBE
          ENDFOR

          FOR COUNTER = 0 TO 5 STEP 1
            START_TOWER.Z = START_TOWER.Z - KUBE
            MOVE_DOWN = START_TOWER
    MOVE_DOWN.Z = MOVE_DOWN.Z + 150

                PTP MOVE_DOWN
            PTP START_TOWER
            CLOSE_GRAB()
                PTP MOVE_DOWN

                MOVE_DOWN = START_KUBE
    MOVE_DOWN.Z = MOVE_DOWN.Z + 150


            PTP MOVE_DOWN
                PTP START_KUBE
                OPEN_GRAB()
                PTP MOVE_DOWN
                START_KUBE.X = START_KUBE.X + KUBE + 8
          ENDFOR


;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP,
2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
;ENDFOLD

END

DEF CLOSE_GRAB ()
          $OUT[3]=TRUE
          WAIT SEC 1.5
          $OUT[3]=FALSE
END

DEF OPEN_GRAB ()
          $OUT[4]=TRUE
          WAIT SEC 1.5
          $OUT[4]=FALSE
END
```