

Курсовая работа

по дисциплине:

«Объектно-ориентированное программирование»

на тему:

«Управление промышленным манипулятором KUKA с
механическим захватом»

Студент:

Шарынин В.А., Кравчук В. А.,
Забегалов А. П., Колобов Я. Д.

Преподаватель:

Ананьевский М.С.

Группа:

3331506/10401

Санкт-Петербург

2024

Оглавление

1	Теоретические сведения	3
1.1	Структура простейшей программы на языке KRL	3
1.2	Переменные и типы данных	4
1.2.1	Простые типы данных	5
1.2.2	Структурные типы данных	6
2	Программирование робота в среде WorkVisual	7
2.1	Подключение SmartPad к компьютеру	7
2.2	Режимы выполнения программы	9
2.3	Определение координат манипулятора	10
3	Программа построения башни	11
	Заключение	13
	Список использованных источников	13
	Приложение 1	13

1 Теоретические сведения

1.1 Структура простейшей программы на языке KRL

Любая программа на языке KRL состоит из двух текстовых файлов с расширениями *.src и *.dat, где * - одно и то же имя. В файле *.src (файл кода) находится код программы, а в файле *.dat (файл данных) расположены описания переменных, точек, массивов и т.д. для сокращения размера файла *.src.

Файл *.src состоит из «главной» и «дополнительных» функций. «Главная» функция должна называться также как и сам (*.src) файл, в котором она находится. 7 Функция на языке KRL состоит из трёх основных разделов – рис 1.1.1.

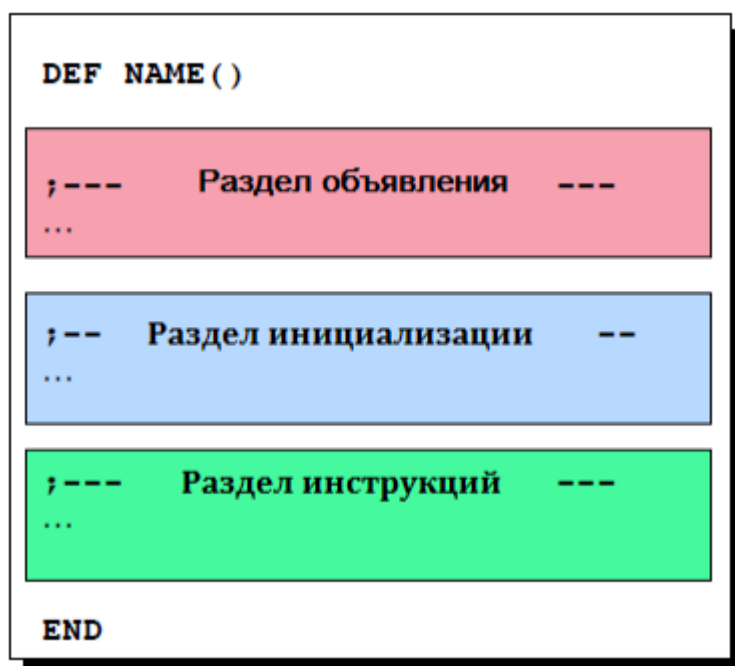


Рисунок 1.1.1 - Разделы функции

Раздел объявления служит для объявления переменных, массивов, точек, структур и т.д. Раздел инициализации нужен для инициализации переменных, массивов и т.д., объявленных в разделе объявления. Раздел

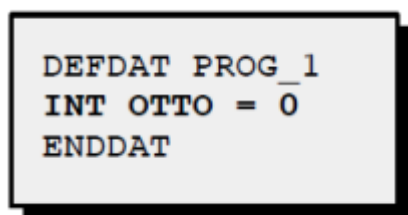
инструкций является основным разделом, в котором находится основной код программы. Объявление функции начинается с ключевого слова «DEF» и заканчивается словом «END».

Например:

```
DEF TEST()  
;Объявление  
;Инициализация  
;Основной код  
END
```

В примере мы видим, что главная функция называется TEST – это значит, что файлы программы называются «TEST.SRC» и «TEST.DAT» соответственно.

Пример файла данных PROG_1.dat приведен на рис. 1.1.2.



```
DEFDAT PROG_1  
INT OTTO = 0  
ENDDAT
```

Рисунок 1.1.2 - Файл данных

Круглые скобки после имени функции сообщают интерпретатору KRL, что ваша программа использует функцию. В разделе 1.10 «Функции и параметры» дана информация о формате записи функций и передаче им параметров (аргументов).

1.2 Переменные и типы данных

В языке KRL существуют как простые, так и структурированные данные.

Для объявления любой переменной желательно использовать ключевое слово «DECL». «DECL» резервирует память под заданную переменную.

Использование данного слова в некоторых случаях не обязательно, но желательно: иначе могут появиться непреднамеренные сбои программы.

1.2.1 Простые типы данных

Под простыми типами данных, подразумевается ряд основных типов данных, которые доступны в большинстве языков программирования, они приведены в таблице 1.2.1.1.

Таблица 1.2.1.1 - 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

Объявление переменных почти ничем не отличается от объявления в других языках программирования за исключением слова «DECL».

Добавим в предыдущий пример несколько переменных. VAR_1 и VAR_2:

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

Здесь переменная VAR_1 имеет тип INT, а VAR_2 тип REAL. Также этот язык поддерживает массивы данных, например:

```
DECL INT MASS[7]  
  
DECL REAL MATRIX[5,4]
```

1.2.2 Структурные типы данных

Если разные типы данных должны быть объединены воедино, то нам на помощь приходит оператор `STRUC`, который позволяет создать новый составной тип данных.

Типичным примером использования составного типа данных является `POS`, объявленный, в файле `$OPERATE.SRC`. Он состоит из шести переменных `REAL` и двух типа `INT`.

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

Например, если будет использоваться переменная `POSITION` типа `POS`, то требуется записать:

```
DECL POS POSITION
```

Присвоить значения отдельным элементам структуры можно с помощью оператора точка «`.`», он позволяет обращаться отдельно к каждому элементу структуры:

```
POSITION.X = 34.4  
POSITION.Y = -23.2  
POSITION.Z = 100.0  
POSITION.A = 90  
POSITION.B = 29.5  
POSITION.C = 3.5  
POSITION.S = 2  
POSITION.T = 6
```

или совместно при помощи «совокупности»:

```
POSITION={X 34.4,Y -23.2,Z 100.0,A 90,B 29.5,C 3.5,S 2,T  
6}
```

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 1.2.2.1 - 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

2 Программирование робота в среде WorkVisual

2.1 Подключение SmartPad к компьютеру

В первую очередь необходимо подключить компьютер к KLI порту контроллера KUKA с помощью кабеля Ethernet. Далее необходимо установить IP-адрес компьютера в регион IP-адреса робота.

Чтобы узнать IP-адрес робота необходимо нажать на значок робота на SmartPad, перейти в раздел Start-up, затем в Network Configuration. IP -адрес робота – 172.31.1.147, поэтому мы установили адрес 172.31.1.148, чтобы он находился в том же регионе, но не совпадал с адресом робота.

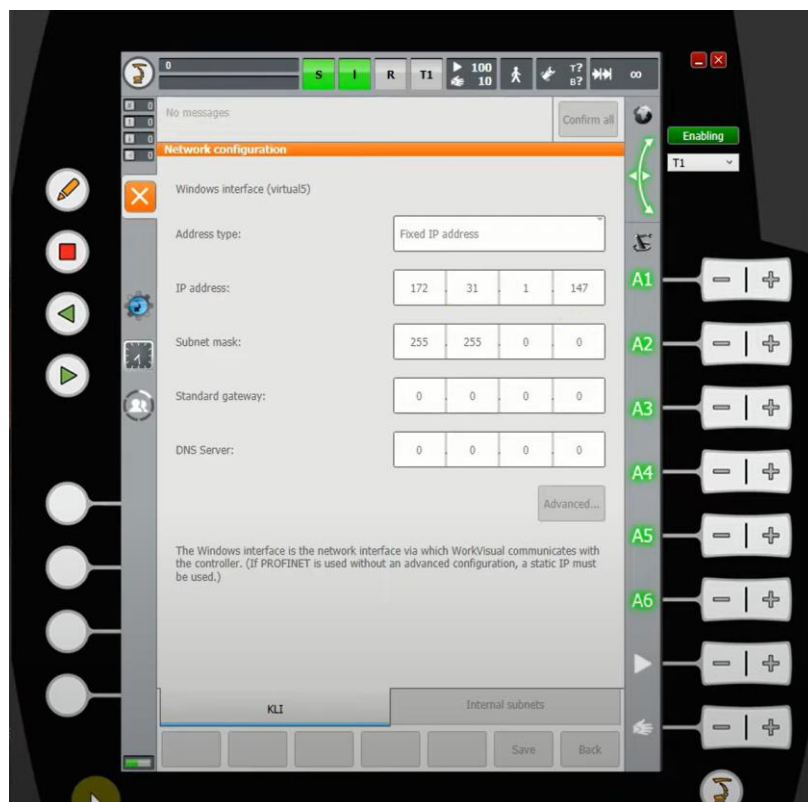


Рисунок 2.1.1 – IP-адрес работа

Помимо этого, для корректной работы с контроллером KUKA, необходимо установить язык системы, такой же как язык системы SmartPad, в нашем случае английский.

WorkVisual позволяет скачивать существующие в работе проекты редактировать, удалять и создавать новые и затем загружать их обратно в память робота. Для того, чтобы можно было принять проекты выгружаемые из WorkVisual необходимо установить на SmartPad уровень доступа Expert. Переходим в раздел Users, устанавливаем режим Expert, используя стандартный пароль KUKA.

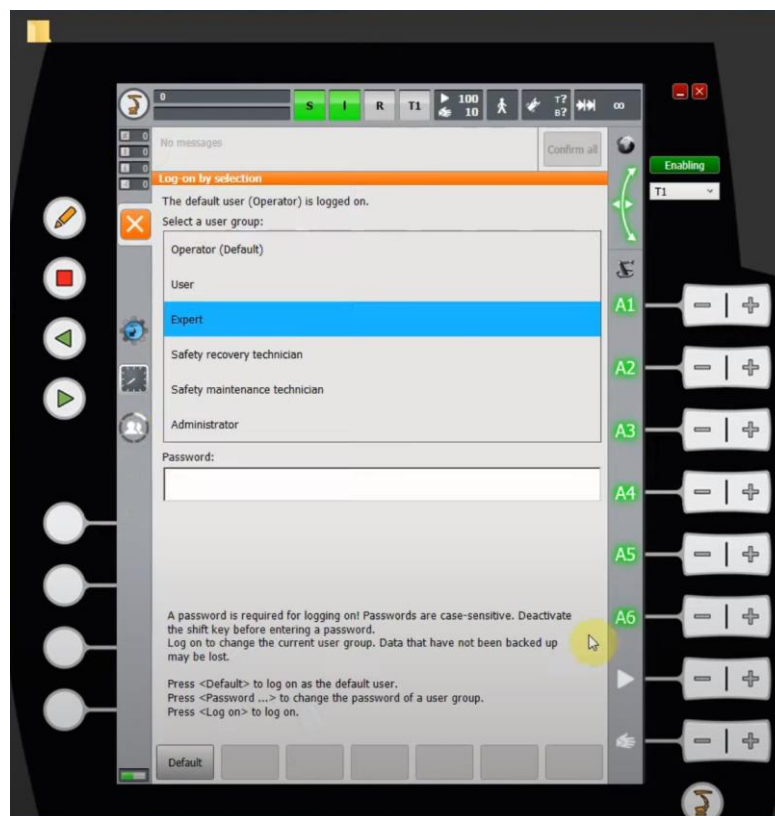


Рисунок 2.1.2 - Установка режима Expert

2.2 Режимы выполнения программы

Робот может выполнять программу в 3 различных режимах, выбор которых осуществляется при помощи диспетчера переключений, расположенного на SmartPad.



Рисунок 2.2.1 - Диспетчер переключений

Режим T1 – ручной снижение скорости. Используется для тестовой эксплуатации, программирования и обучения. Имеет пониженные скорости

Режим T2 – ручной высокая скорость. Используется для тестовой эксплуатации. Скорость соответствует запрограммированной.

Режим AUT – автоматический режим. Используется для нормальной эксплуатации.

В своей работе мы использовали режим T1 для тестов программы и режим AUT для демонстрации выполнения программы.

2.3 Определение координат манипулятора

Для определения координат манипулятора, чтобы выставить далее от этой точки базу необходимо посмотреть эти координаты на SmartPad.

Config->Display->Actual position

Далее параллельно с ходом программы и ручным движением манипулятора будут показаны координаты манипулятора относительно центра крепления манипулятора к столу.

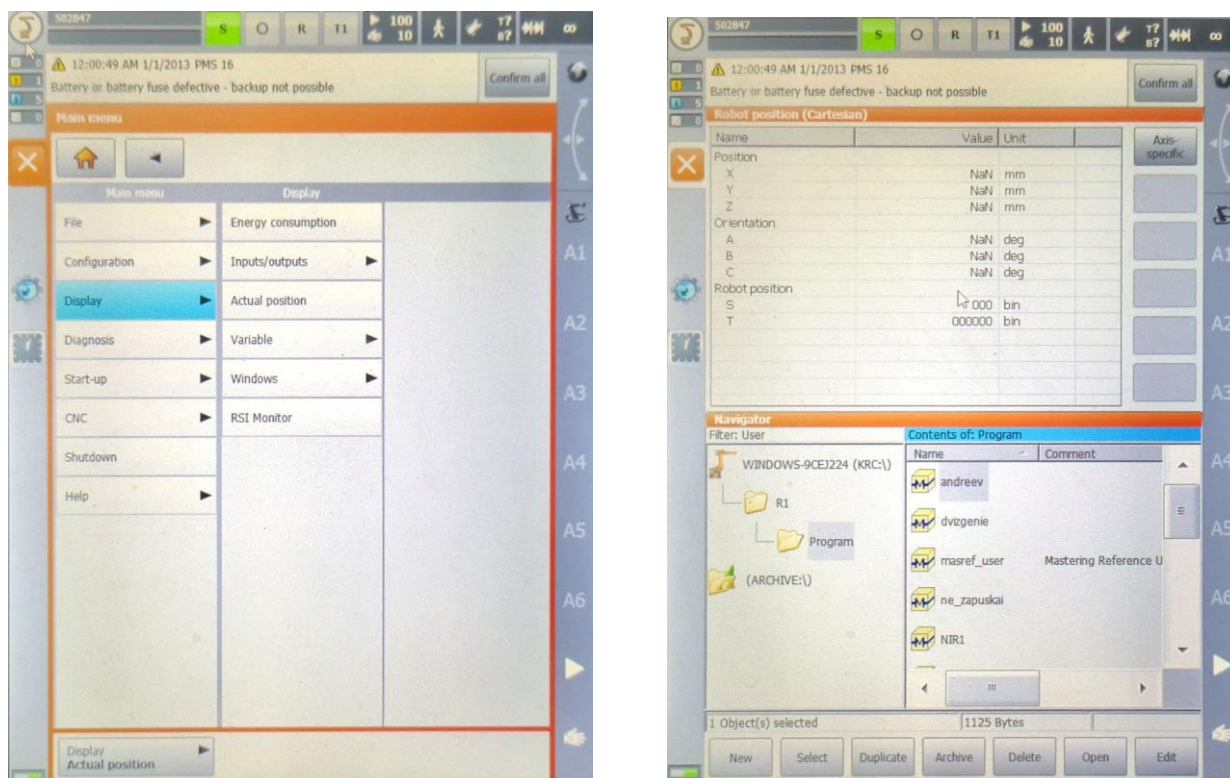


Рисунок 2.3 – Определение позиции захвата

3 Программа построения башни

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
```

В данном блоке (Блоке объявления) объявляется функция, а также переменные и структуры, которые будут использоваться в дальнейшем.

Обязательно все декларации переменных должны быть выполнены перед блоком инициализации.

Frame - это системно определенная структура, которая определяет точку по 6-ти координатам X, Y, Z, OZ, OY, OX.

```
;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
```

Здесь мы уже начинаем определять размер кубика, от какой базы работаем в данный момент, а также координаты начала башни и начало линии расположения кубиков.

```
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

```

Так выглядит цикл для построения башни. Чтобы не сбивать кубики используется заход на них сверху. Это реализуется в точке MOVE_DOWN. Чтобы построить башню из большего количества кубов необходимо лишь увеличить число в цикле FOR.

Функция разборки башни реализуется аналогичным образом, но в обратном порядке.

```

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

```

Отдельное слово хочется сказать про управление захватом. Захват - это дополнительный модуль манипулятора, работа которого была встроена в манипулятор путем анализа выходных значений с OUT[3] и OUT[4]. При подаче логической единицы на OUT[3] захват будет закрываться и если это будет происходить достаточно долго (примерно 2 сек.), то захват перестанет работать и придется перезагружать всего робота. Поэтому эмпирическим методом выведено, что оптимальным временем для полного раскрытия и закрытия захвата необходимо 1,5 сек.

Открытие захвата происходит соответственно при подаче «1» на OUT[4].

Заключение

В ходе выполнения работы были произведены установка и настройка программной среды Work Visual. Была произведена корректировка базы работа и написана программа для построения башни из кубов с использованием, установленного на манипулятор, захвата. Также были проведены успешные тесты его работы на маленьких скоростях и на больших рабочих.

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 1

```
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
```