Санкт-Петербургский политехнический университет Петра Великого

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](#_Toc165980516)

[1.1 Структура простейшей программы на языке KRL 3](#_Toc165980517)

[1.2 Переменные и типы данных 4](#_Toc165980518)

[1.2.1 Простые типы данных 5](#_Toc165980519)

[1.2.2 Структурные типы данных 6](#_Toc165980520)

[2 Программирование робота в среде WorkVisual 7](#_Toc165980521)

[2.1 Подключение SmartPad к компьютеру 7](#_Toc165980522)

[2.2 Режимы выполнения программы 9](#_Toc165980523)

[2.3 Определение координат манипулятора 10](#_Toc165980524)

[3 Программа построения башни 11](#_Toc165980525)

[Заключение 13](#_Toc165980526)

[Список использованных источников 13](#_Toc165980527)

[Приложение 1 13](#_Toc165980528)

# Теоретические сведения

## Структура простейшей программы на языке KRL

Любая программа на языке KRL состоит из двух текстовых файлов c 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 трѐх основных разделов – рис 1.1.1.

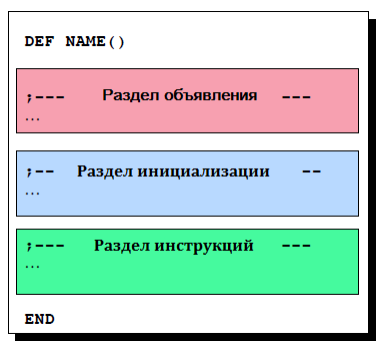
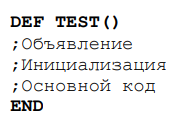


Рисунок 1.1.1 - Разделы функции

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:



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.

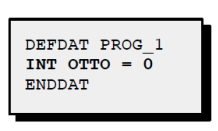


Рисунок 1.1.2 - Файл данных

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).

## 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.

### Простые типы данных

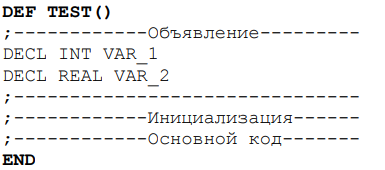
Под простыми типами данных, подразумевается ряд основных типов данных, которые доступны в большинстве языков программирования, они приведены в таблице 1.2.1.1.

Таблица 1.2.1.1 - Простые типы данных

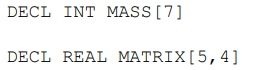


Объявление переменных почти ничем не отличается от объявления в других языках программирования за исключением слова «DECL».

Добавим в предыдущий пример несколько переменных.VAR\_1 и VAR\_2:



Здесь переменная VAR\_1 имеет тип INT, а VAR\_2 тип REAL. Также этот язык поддерживает массивы данных, например:



### 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.

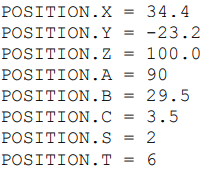
Типичным примером использования составного типа данных является POS, объявленный, в файле $OPERATE.SRC. Он состоит из шести переменных REAL и двух типа INT.



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



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



или совместно при помощи «совокупности»:



Эти две записи выполняют одинаковую инициализацию.

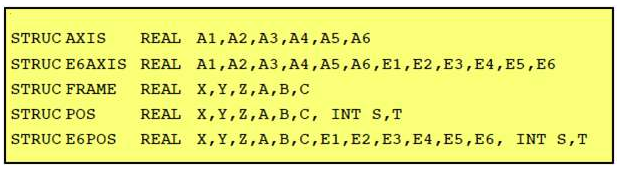
Также тип данных может указываться в начале инициализации структуры.

Например:



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

Таблица 1.2.2.1 - Типы структур



# Программирование робота в среде WorkVisual

## 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.

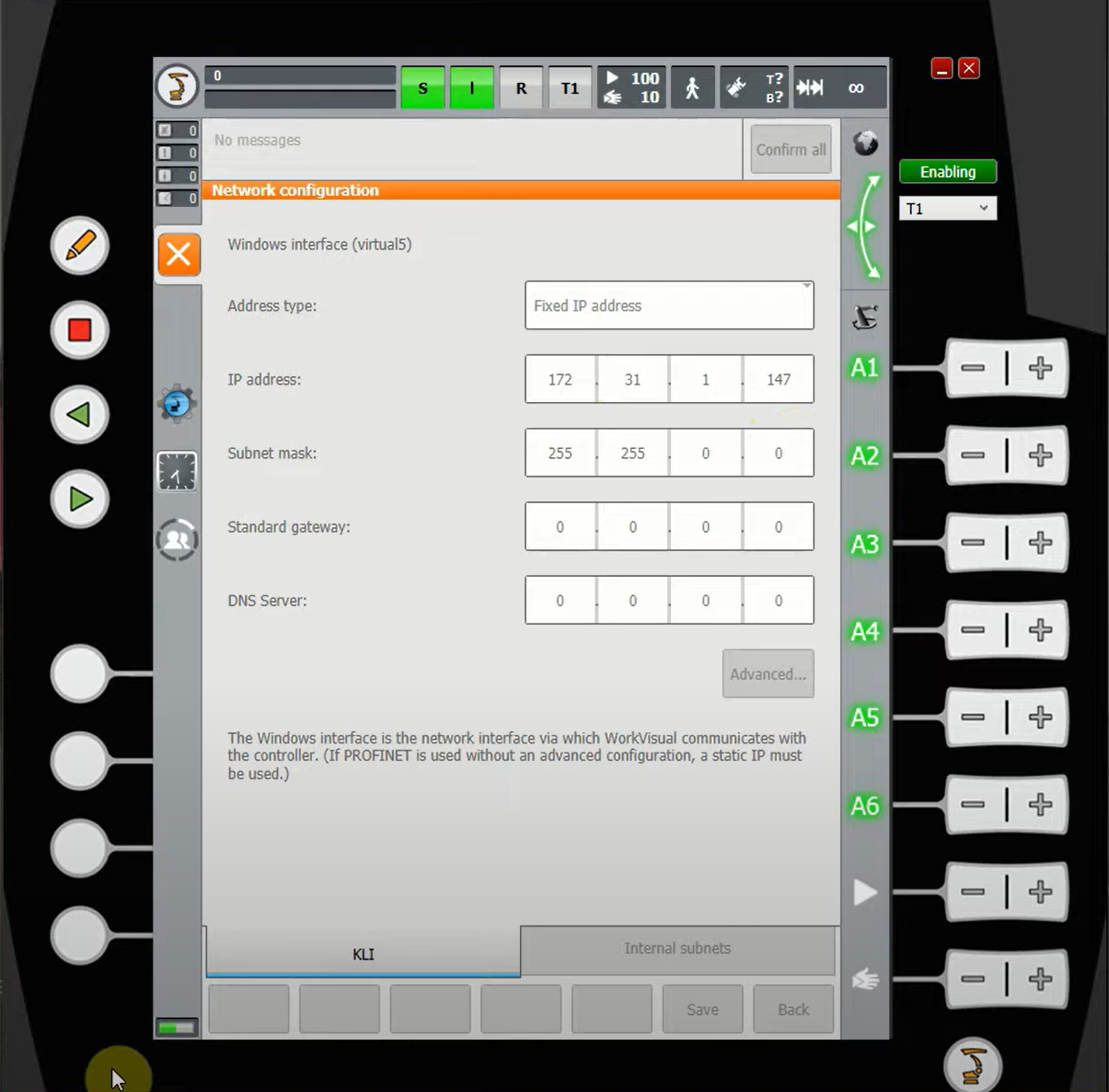


Рисунок 2.1.1 – IP-адрес робота

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 роботе 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.

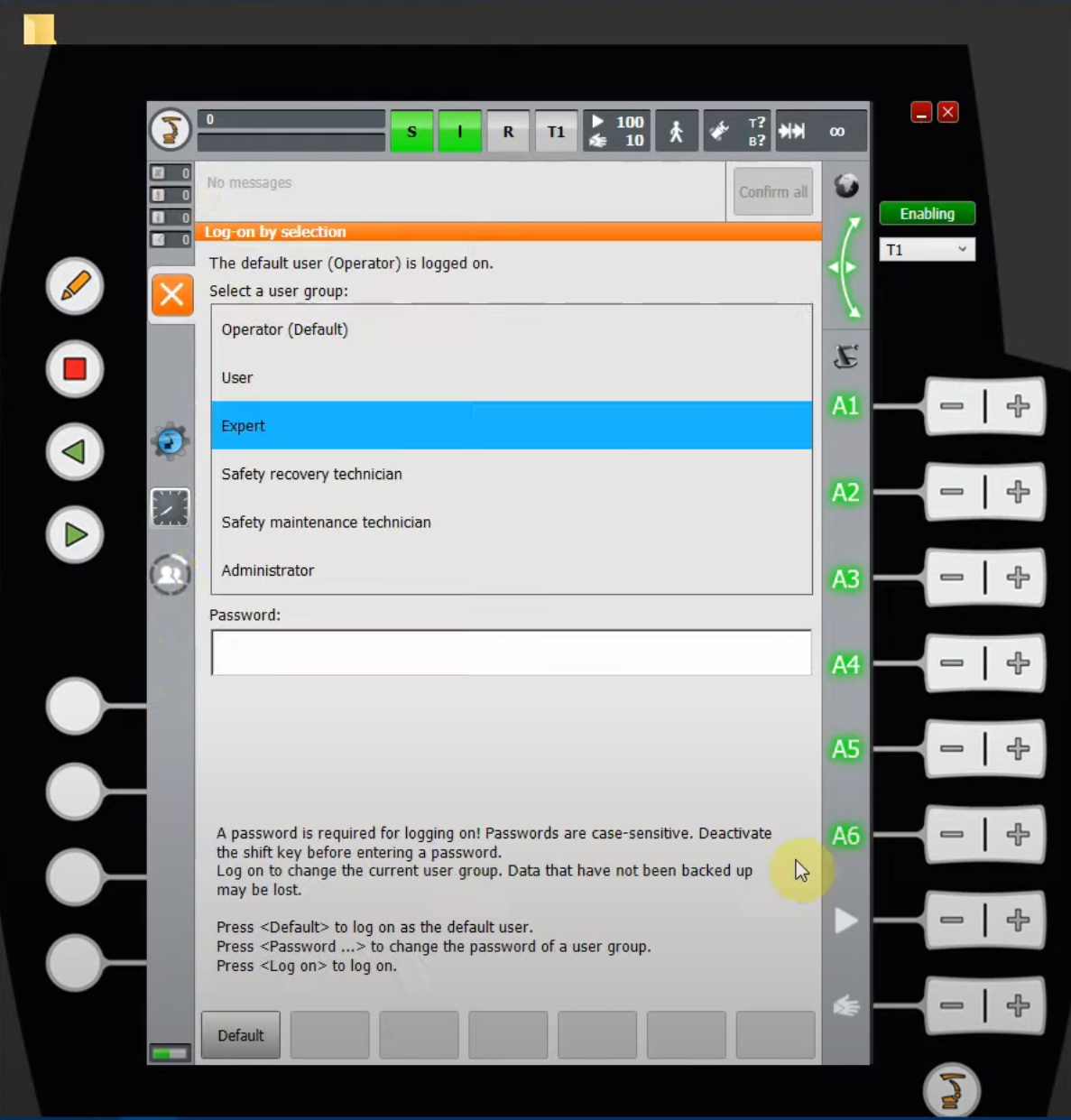


Рисунок 2.1.2 - Установка режима Expert

## 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 Т1 – ручной снижение скорости. Используется для тестовой эксплуатации, программирования и обучения. Имеет пониженные скорости

Режим Т2 – ручной высокая скорость. Используется для тестовой эксплуатации. Скорость соответствует запрограммированной.

Режим AUT – автоматический режим. Используется для нормальной эксплуатации.

В своей работе мы использовали режим Т1 для тестов программы и режим AUT для демонстрации выполнения программы.

## Определение координат манипулятора

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

Config->Display->Actual position

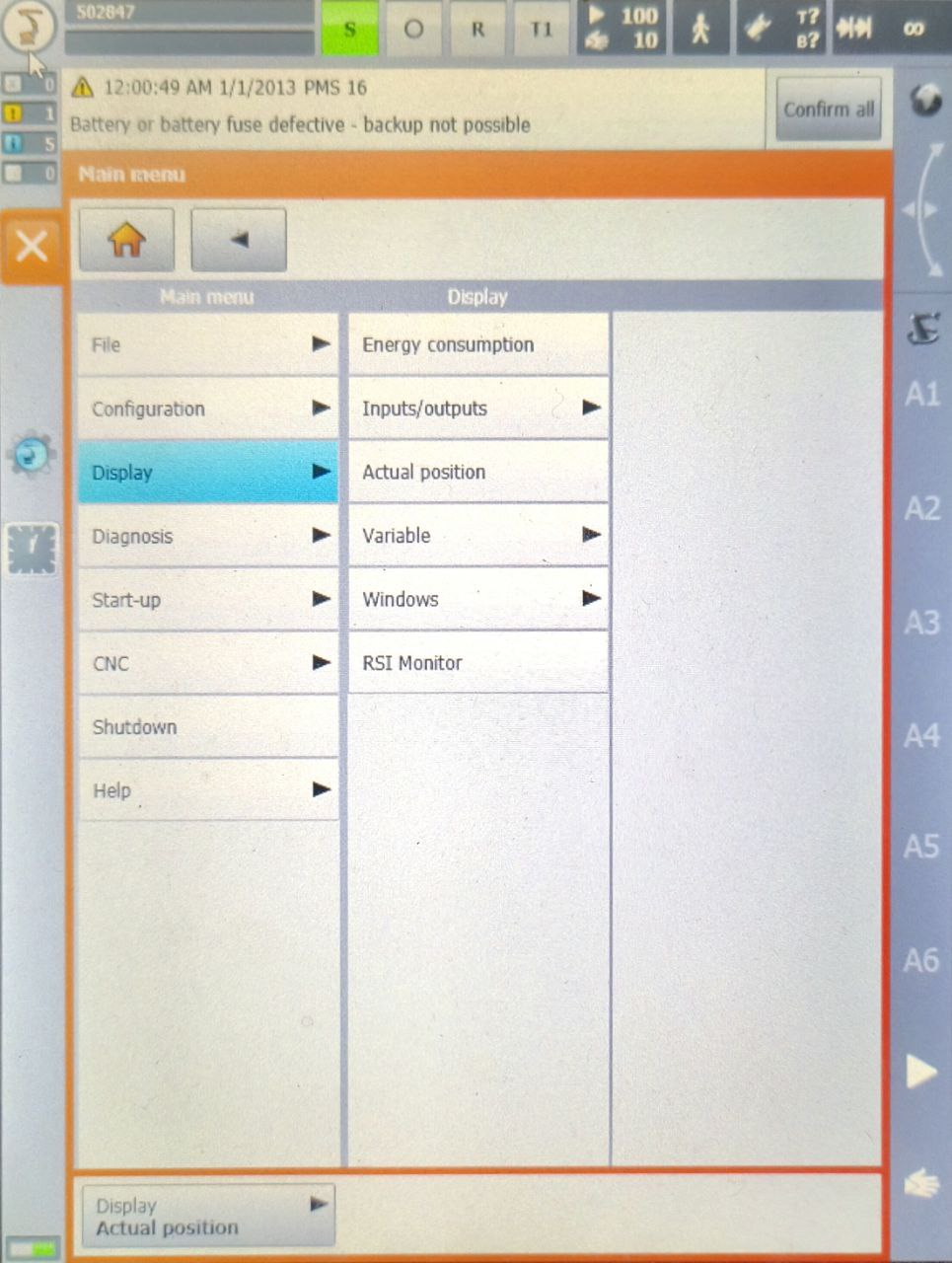
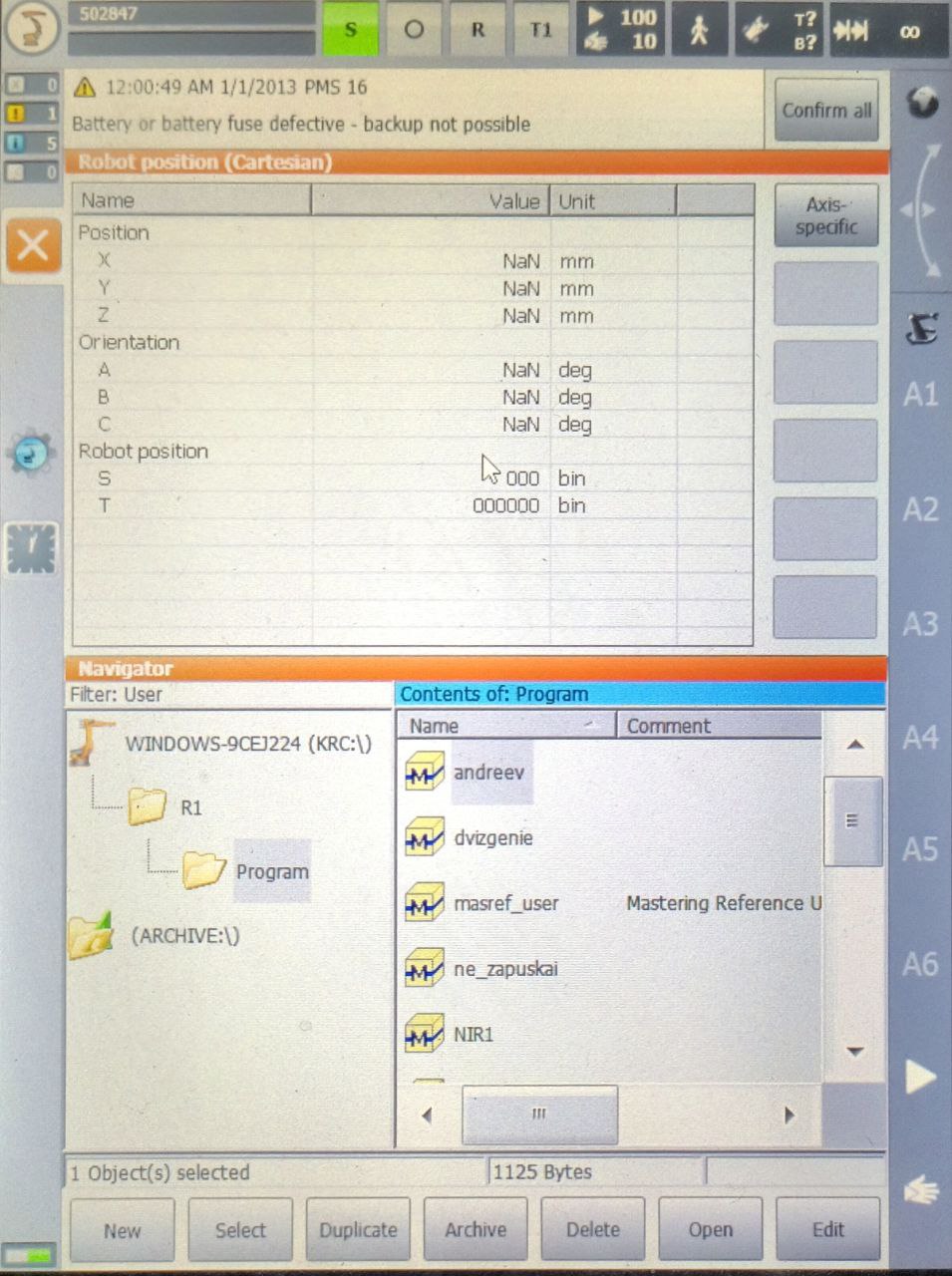
Далее параллельно с ходом программы и ручным движением манипулятора будут показаны координаты манипулятора относительно центра крепления манипулятора к столу.

Рисунок 2.3 – Определение позиции захвата

# Программа построения башни

Веь 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 робота 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