# Education Cell
## Student Exercises

FANUC


FANUC EDUCATIONAL PACKAGE

**FA**
CNCs, Servo Motors and Lasers

**ROBOTS**
Industrial Robots, Accessories and Software

**ROBOCUT**
CNC Wire-Cut Electric Discharge Machines

**ROBODRILL**
Compact CNC Machining Centres

**ROBOSHOT**
Electric CNC Injection Moulding Machines

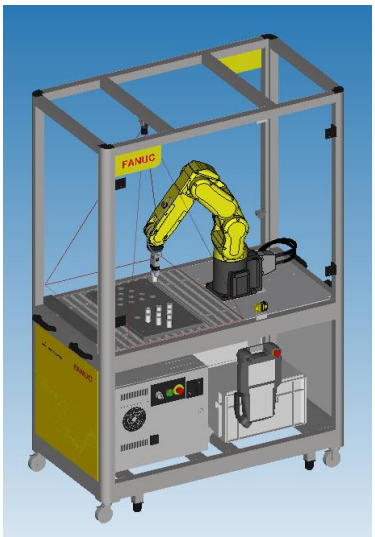
FANUC

# Student Exercises

## Table of contents

## Abstract

In this exercise, we will give you a quick overview of the basic controls and knowledge needed for every other steps and exercises on the robot.
After this exercise, you should be able to turn the robot on and off, switch between the different modes and do some basic jogging.




**PLEASE NOTE THAT**

All exercise references are referred to:
**R-30*i*B basic operator manual [B-83284EN_05]**

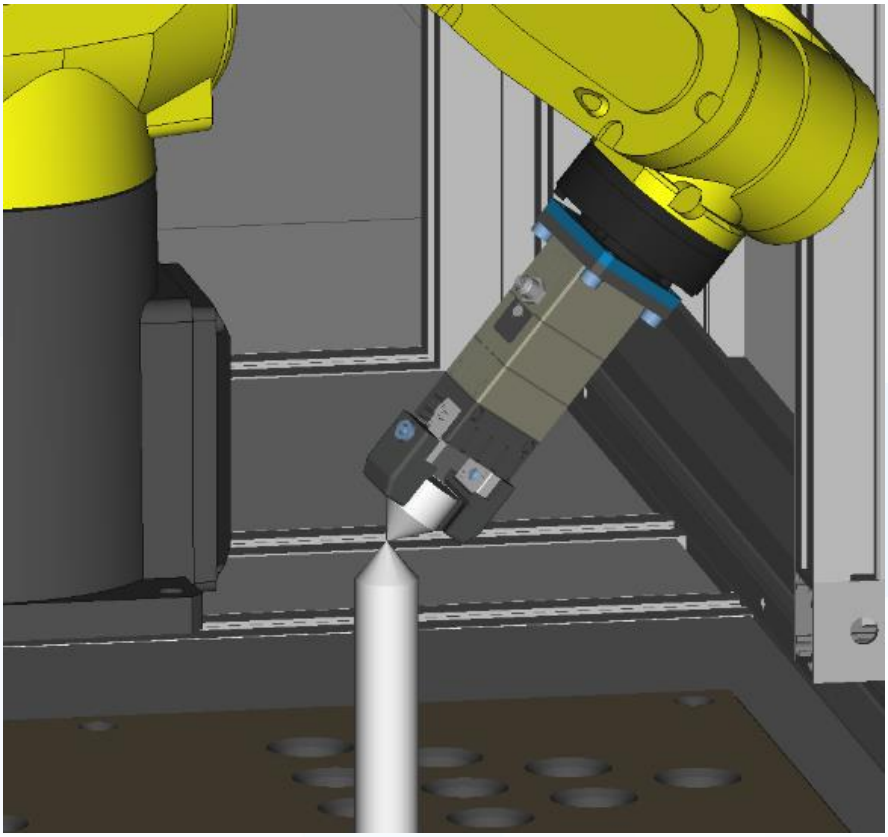
In further exercises, we will just refer to it as Operator Manual. (OM Section…)

Every exercise is based on knowledge acquired in previous exercises. This means you should make sure that all the knowledge in the exercise is known before continuing. If something in the recapitulation is not known, or not very clear, you should do that exercise again.

In the beginning of every exercise, we give you a list of equipment, you need. You should make sure that you have got all the needed equipment, because otherwise, you will not be able to do every part of the exercise.

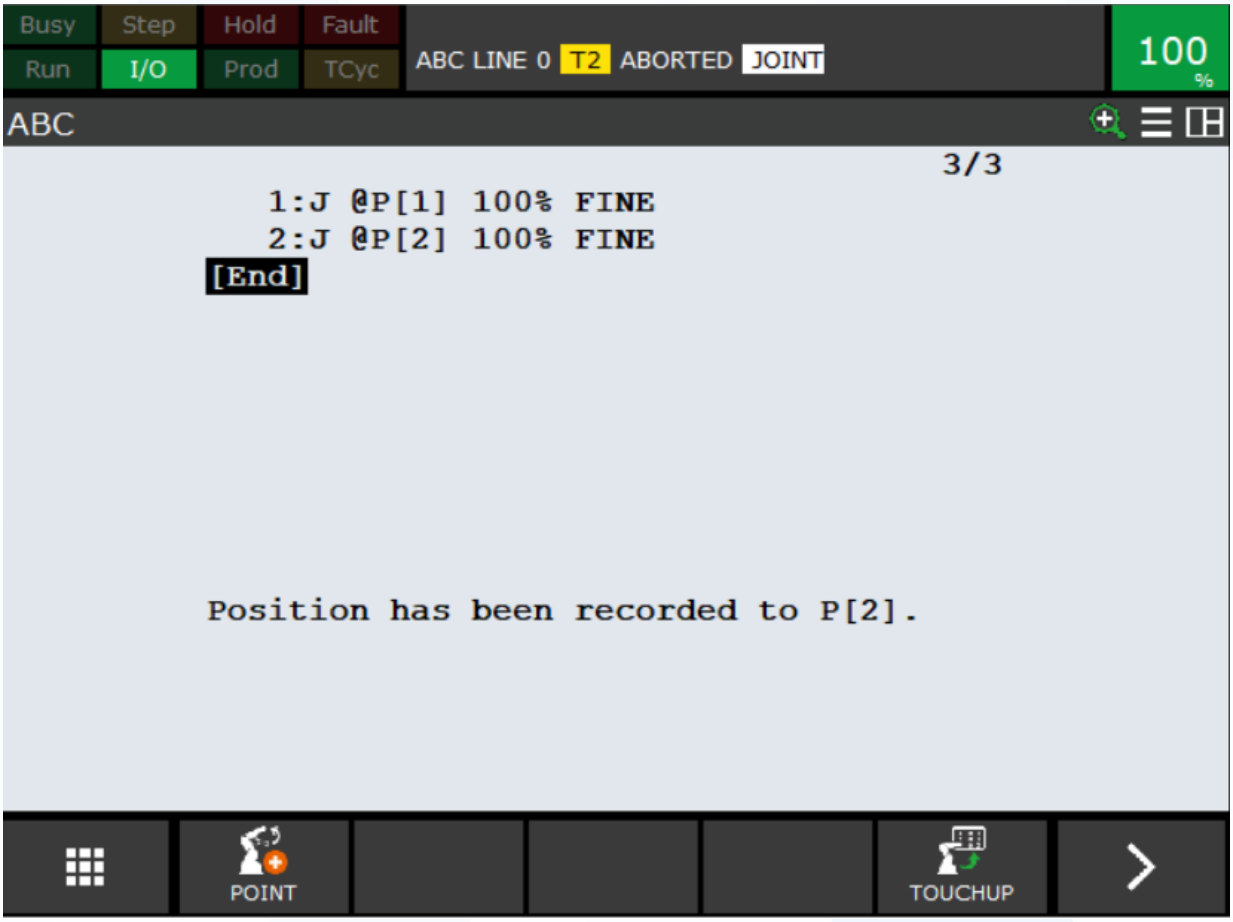
## Abstract

In this exercise, we want to show you the concept and the setup of TCP.
TCP stands for Tool Centre Point and is especially important for an easier way to jog the robot in tool coordinate system.
But not only in jogging there can be a difference. Also when changing the attitude of the final flange (and therefor of the tool) while moving, the tool will not move in a linear motion and the speed will be different from the one of the flange.
At the end of this exercise, you should be able to understand the concept of TCP and the use of this setting, and you should be able to set a TCP with all the different methods available.



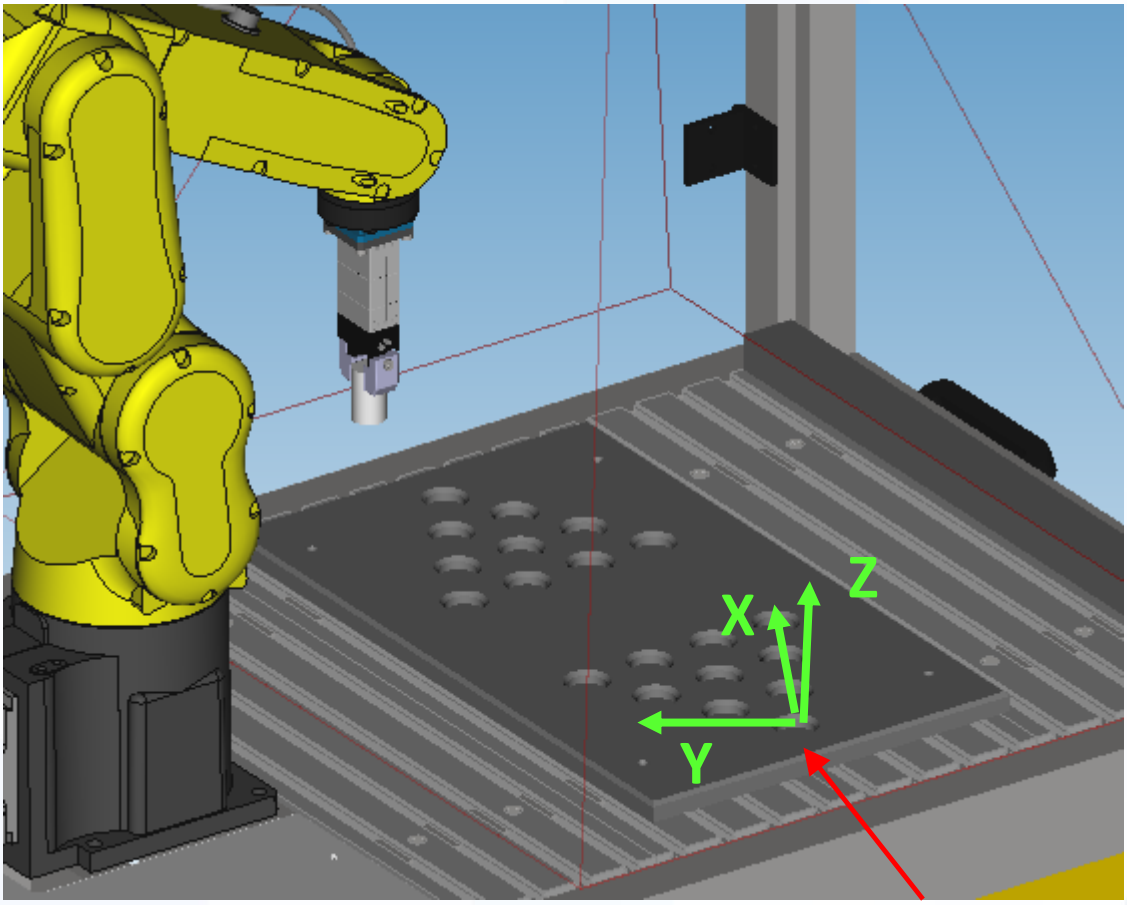
## Abstract

In this exercise you should learn the basics about programming a robot and the different options available to program a robot.
After this exercise, you should be able to program a simple motion and put comments into your program.



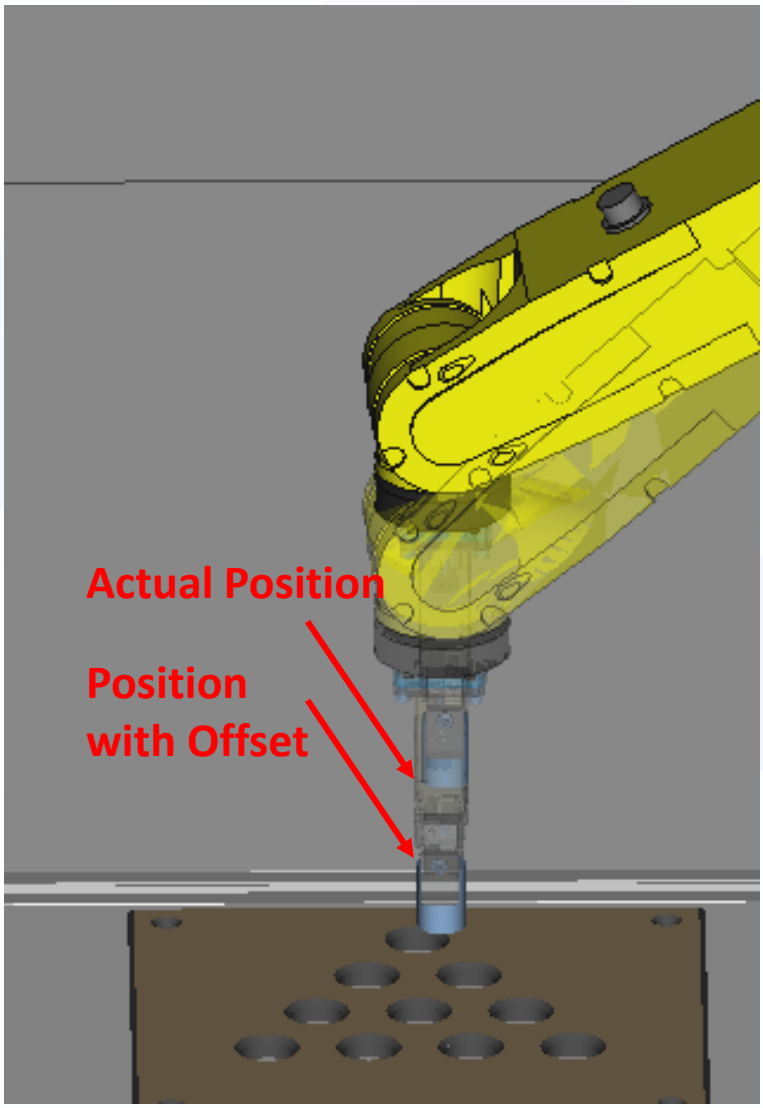
## Abstract

In this exercise, you learn the basics of User Frame teaching and using.
User Frame (UFrame) is a Cartesian coordinate system that you set up for an easier way to teach and operate your robot.
After you did this exercise, you should be able to put in place a UFrame of your choice and do programs with that UFrame.
You should also be able to adjust your program to use it on different UFrames and know how to program to be able to rapidly and without much work change between several UFrames.



## Abstract

In this exercise, you should learn the concept and use of the offset function.
You should learn how to program an offset into the Position Register(PR) of the robot an use this offset later in your program.
You should be able to program without using direct position teaching, while using the PR for positioning and offset of your tool, which should make your programing more effective, faster and also easier.



## Abstract

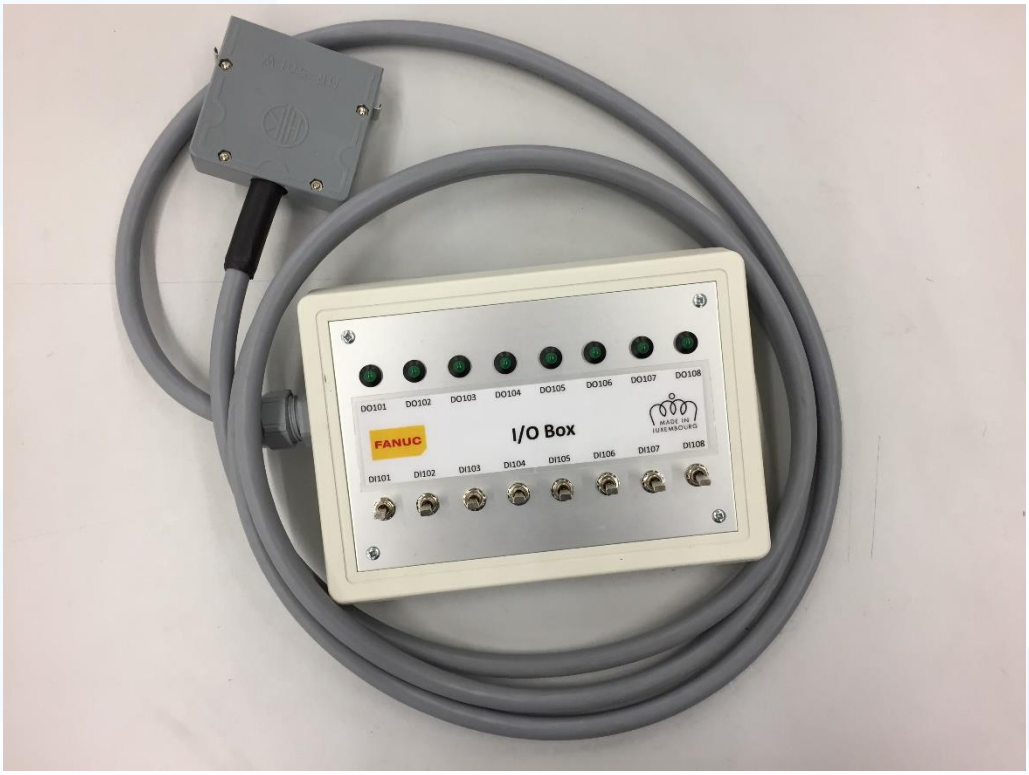
In this exercise, we give you a more thorough overview of the different options and possibilities that you can use to program your robot.
Using a simple program that we will change over the course of this exercise, we want to show you all the different options and get you to think about creating your own “more complicated” programs.
It is very important that you have done exercise 3 with success, because this exercise builds on knowledge acquired in exercise 3.
In this exercise we put all the “solutions” right behind the instructions. You should however try and do the exercise without looking at those “solutions”. If you can’t, do the exercise with them and then try and do it again without.
Also, try and put everywhere comments on what you have programmed to make reading your program easier for other people. This is not very important for this exercise, but as it is an exercise, you should also practise this.

```
ADVANCED_PROG_EX_2                                    1/12
    1:  !UFrame Set
    2:  UFRAME_NUM=1
    3:  !Loop Instruction
    4:  FOR R[32:ForTo Value Ex]=1 TO 10
    5:  !Setting R33
    6:  R[33:If Value Ex]=
     :  R[32:ForTo Value Ex] MOD 2
    7:  !If Commands
    8:  IF R[33:If Value Ex]=0,
     :  CALL ADVANCED_PROGRAM_PART1
    9:  IF R[33:If Value Ex]=1,
```

```
ADVANCED_PROG_EX_2                                   12/12
    5:  !Setting R33
    6:  R[33:If Value Ex]=
     :  R[32:ForTo Value Ex] MOD 2
    7:  !If Commands
    8:  IF R[33:If Value Ex]=0,
     :  CALL ADVANCED_PROGRAM_PART1
    9:  IF R[33:If Value Ex]=1,
     :  CALL ADVANCED_PROGRAM_PART2
   10:  !End of For loop
   11:  ENDFOR
[End]
```

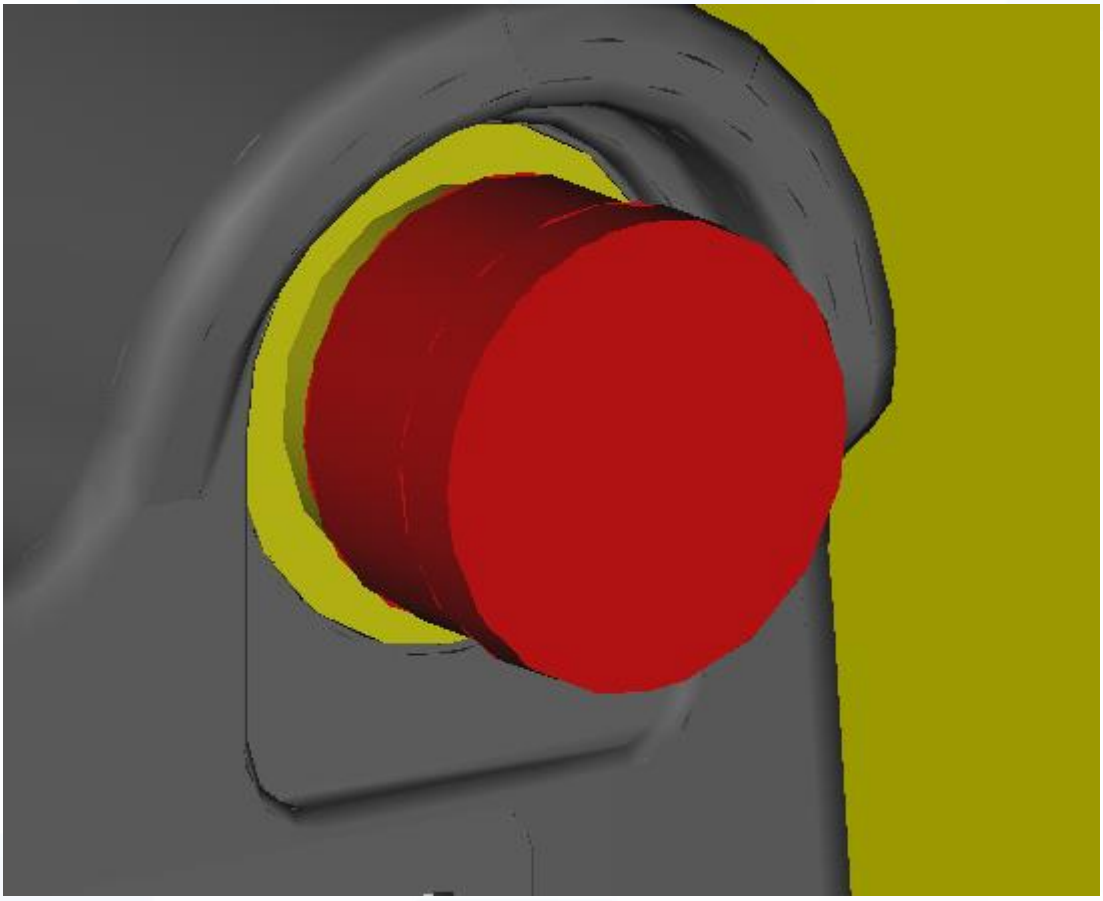
## Abstract

In this exercise, you are going to learn about Input and Output (I/O) of the robot.
I/O is nothing more than a simple mean to communicate through signals with other devices.
I/O is very important if the robot does not operate on it's own, but with peripheral devices, such as detectors, cameras, other robots etc.
To simulate other devices, we designed an I/O box.
This exercise can not be made without those boxes so we strongly advise you to firstly build those boxes and then do this exercise. The instructions and parts list needed are all included in the appendix of this exercise.
To simulate those inputs and outputs, we will use switches as inputs and LEDs as outputs to visualise them.



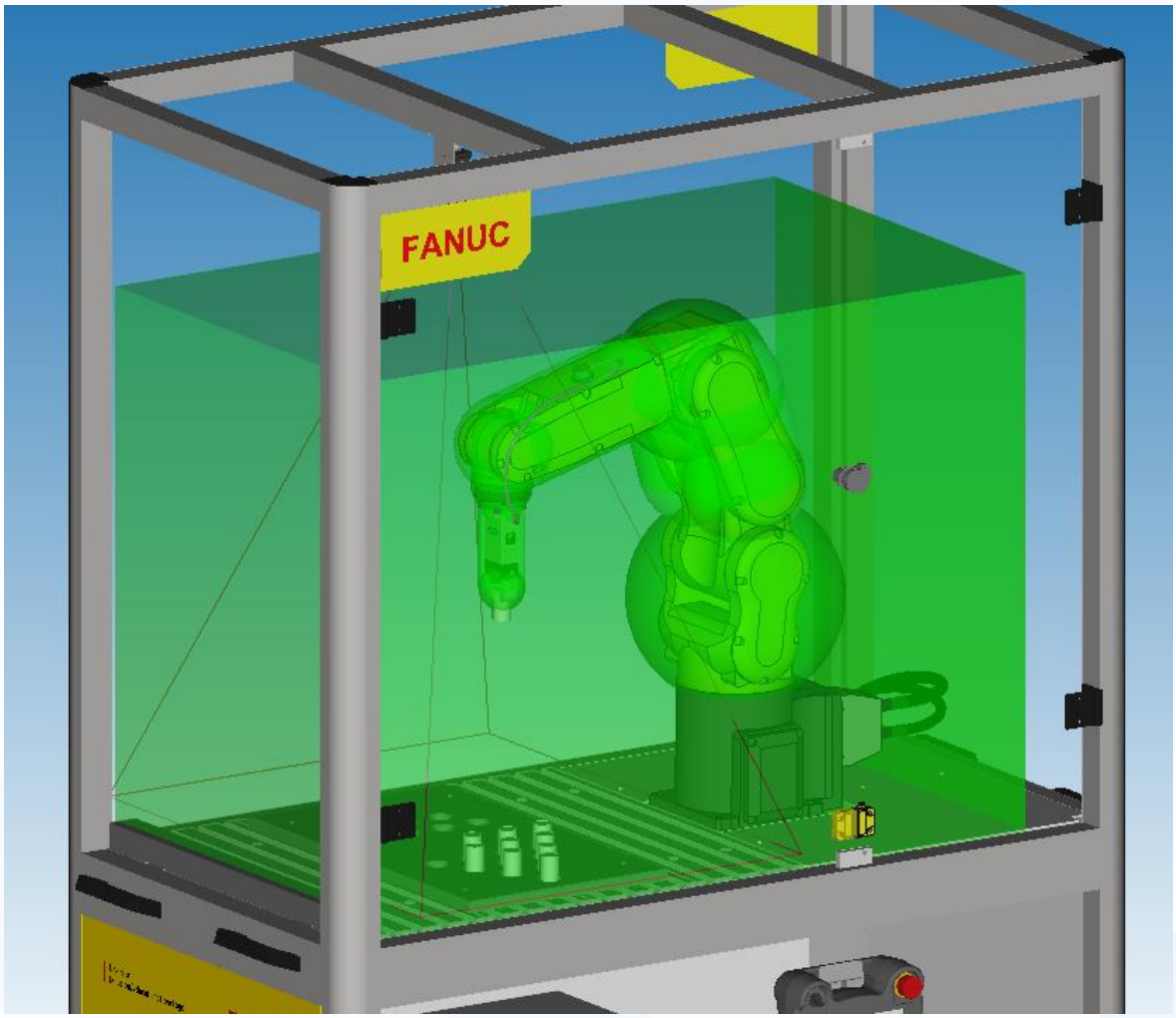
## Abstract

This exercise is kind of an overview over the different stops that are available on FANUC robots, the reasons why we use the different stops in different cases and norms that are behind those stops.
You should learn over the period of this exercise the differences of the stops and when which stop will occur and understand why a certain stop is used.



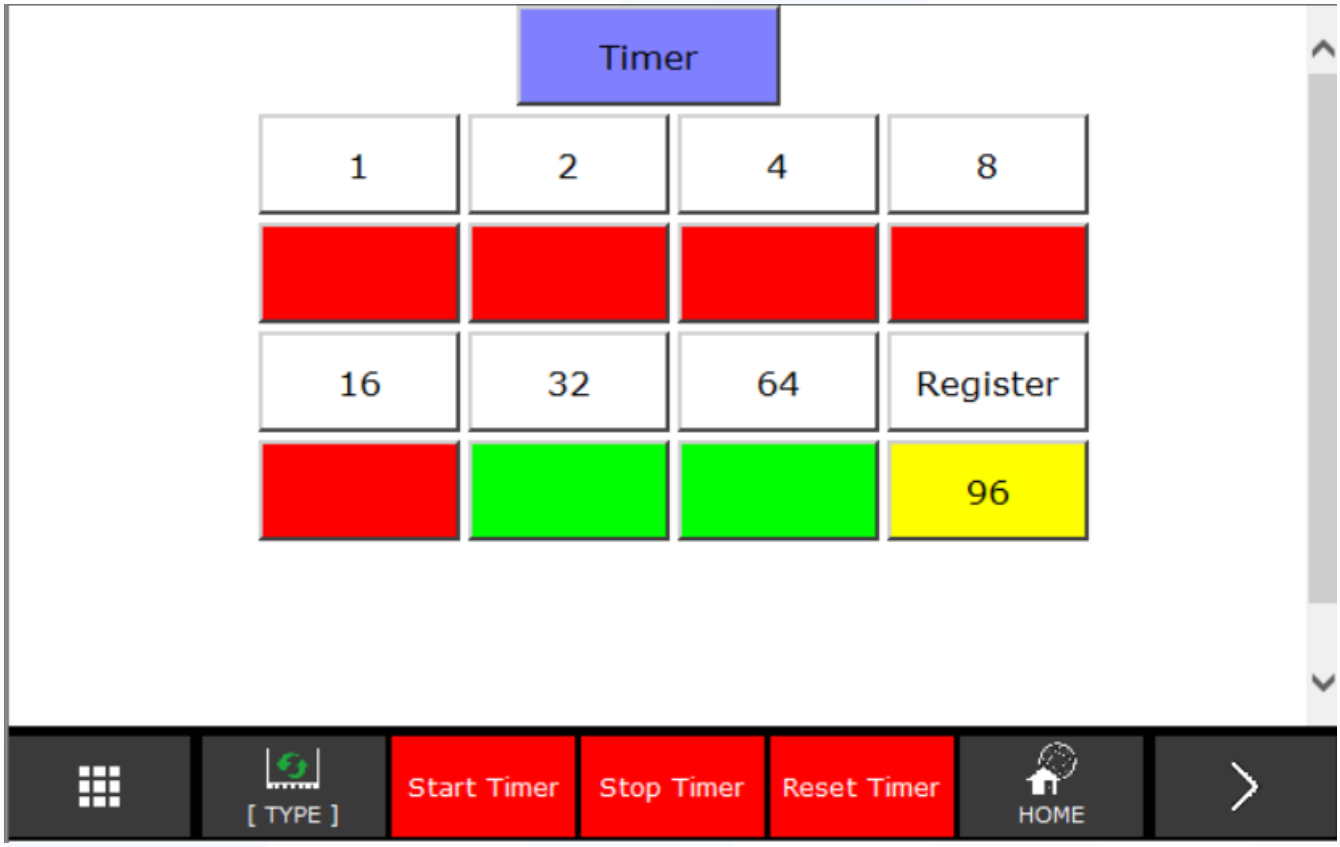
## Abstract

In this exercise, you will learn the principles of DCS Safe Zone and learn how to set up a DCS Safe Zone on your own.
DCS Safe Zone is an virtual box. You can either allow the robot to move exactly in that virtual box, or tell the robot that it is prohibited to move inside that box.
So DCS Safe Zones are used to restrict the movement of the robot. But it does not have to be only a box, it can also be a more complex shape.



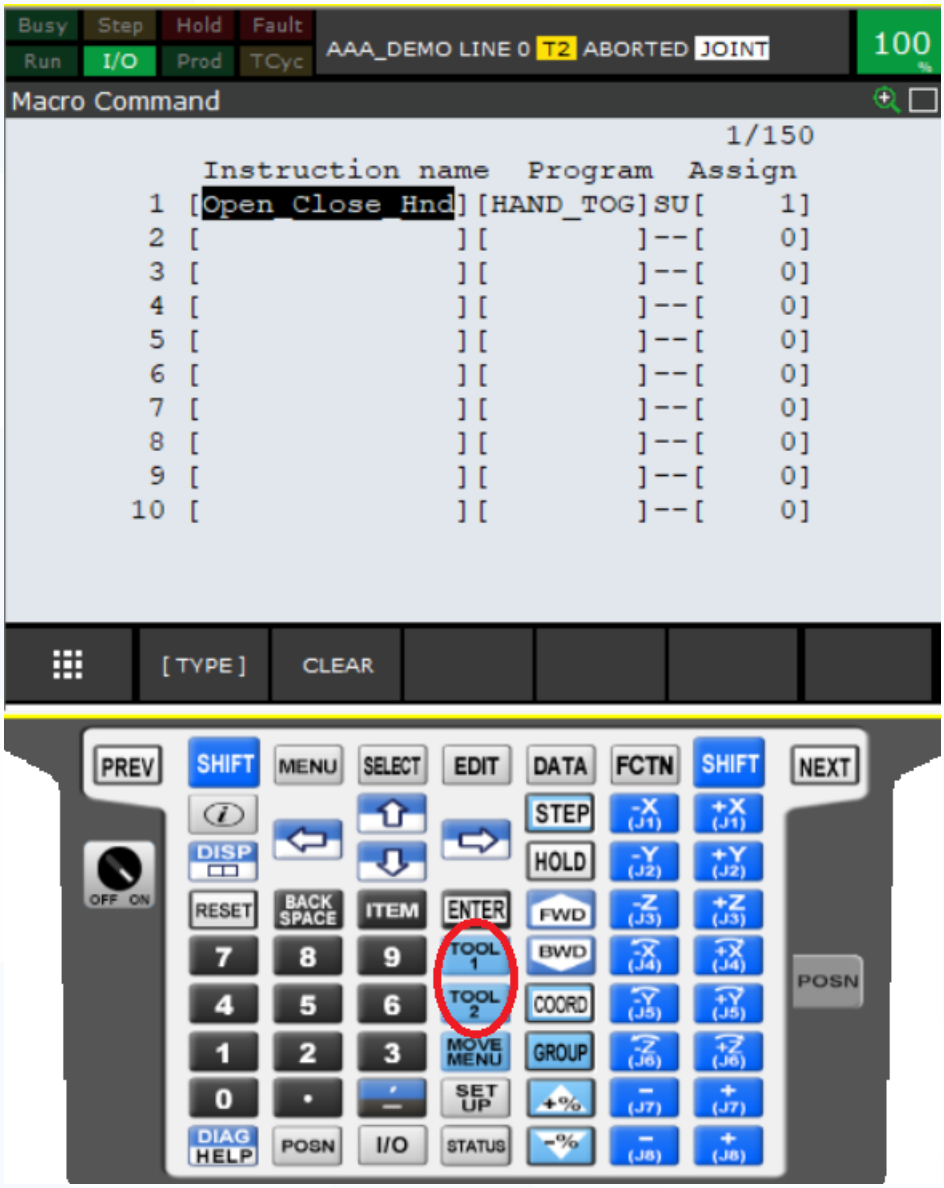
## Abstract

In this exercise, you should learn how to customize the screen of the iPendant (Teach Pendant) and how to call those screens in programs.
You should also learn how the different controls available for customization.
After this exercise, you should be able to create, upload and use a html site for the TP.



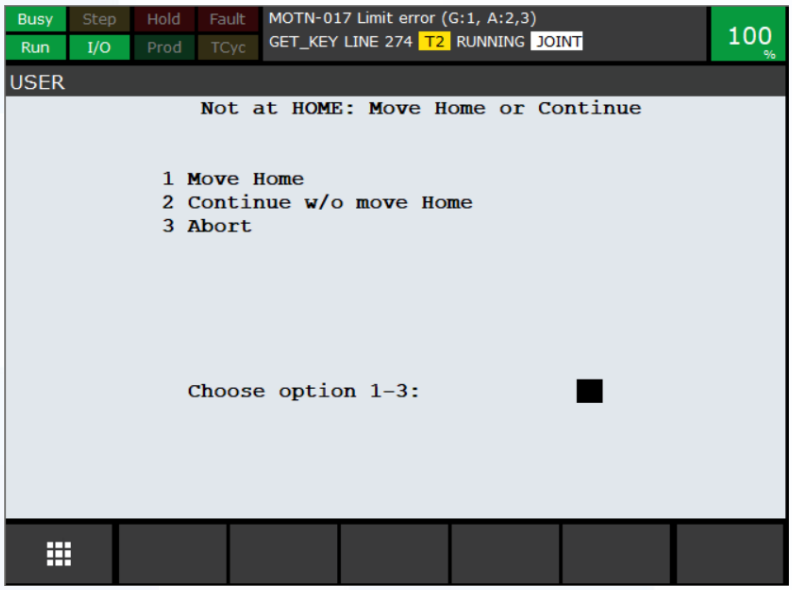
## Abstract

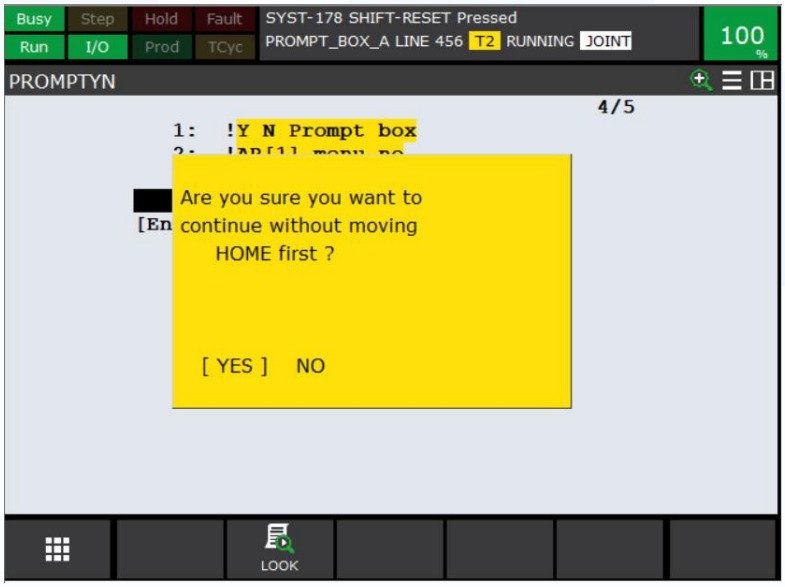
In this exercise, we want to show you the creation and the set up of a macro.
A macro is a program you can call on the Teach Pendant with the “User” keys. A Macro can also be called in other programs.



## Abstract

Menu utility are macros you can use in programs. The macros are special because they are like pop ups in the program where you have to press a key to continue.
In this exercise, we will see 3 different types, the PROMPTOK, PROMPTYN and LISTMENU.



FANUC

Exercise 1

# Introductory Exercise

FANUC

Exercise 1

# Introductory Exercise

## Table of contents

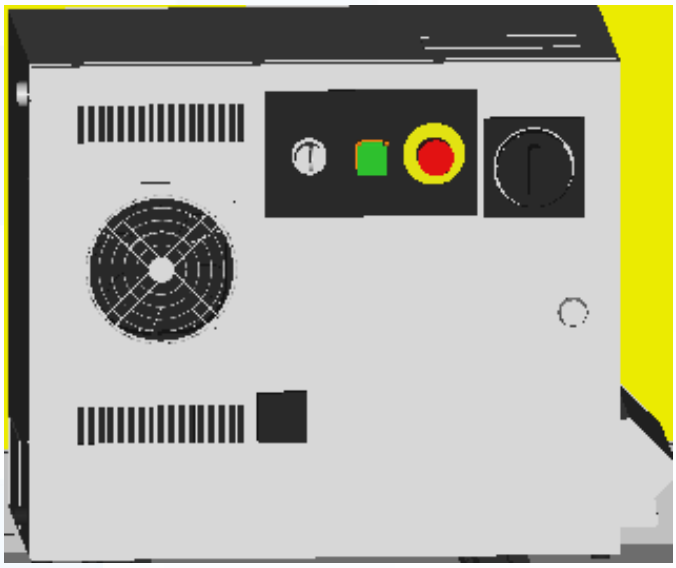
## Equipment

For this exercise, you need:

Education Cell

## Robot Controller

The first thing that we start with is the controller.



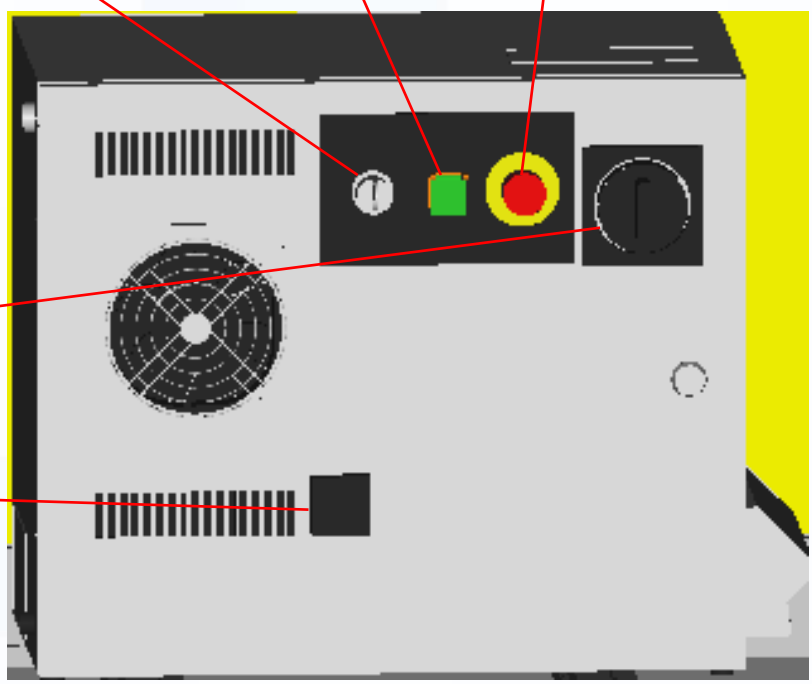
The controller controls the robot, houses the hardware and software and is in a way the “brain” of the robot.
On the following page we will go through the different keys and switches on the controller.

## Robot Controller

Cycle start button, is used to start a cycle.

Emergency stop Button

Three-Mode Switch (for more details, refer to OM section 5.2.2 “Three-Mode Switch”)



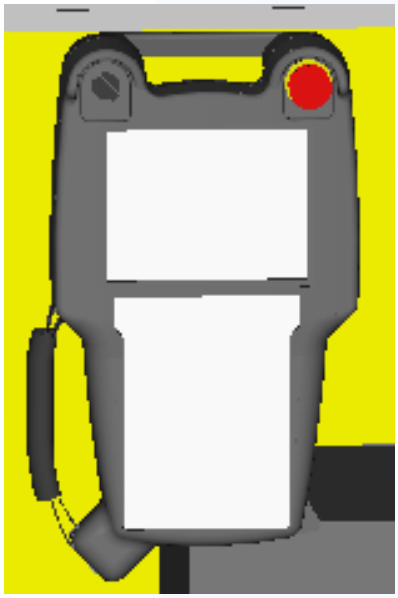
Power switch

USB Port

## Teach Pendant

The Teach Pendant is used to operate and program the robot. It is the interface between you and the controller.
All programming and testing is being done with the Teach Pendant.

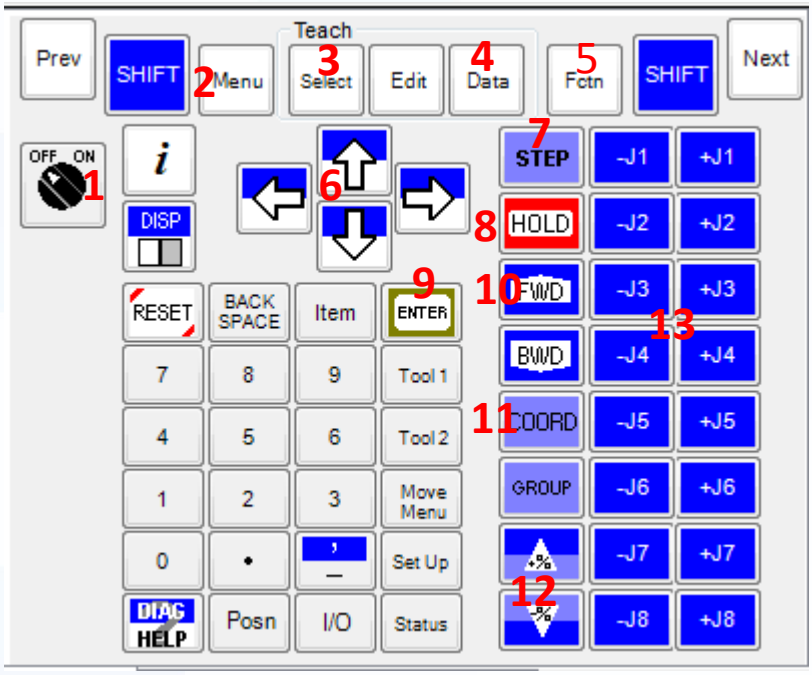
For more details refer to OM section 2.3.1 “Teach Pendant”.



For a more detailed function overview of the Teach Pendant look at the following page.

## Teach Pendant

Most Important features of the Teach Pendant:




1. Teach Pendant Enable switch enables Teach Pendant
2. Menu key displays the screen menu.
3. Select key enters program list.
4. Data key enters register list.
5. Function key displays the function menu.
6. Cursor keys move the cursor.
7. Step key enables/disables step mode. (refer to OM section 6.3.2 “Step Test”)
8. Hold key holds the program.
9. Enter key enters selected item.
10. Forward key forwards program to next line.
11. Coordinate key changes the coordinate system currently selected.
12. Override key changes the override.
13. Jog keys allow to jog the robot in teach mode.

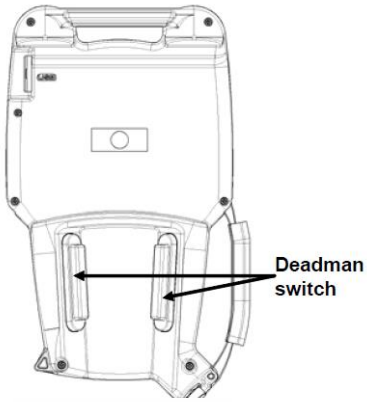
## Robot

The robot mechanical unit consists of several limbs connected to each other with axes on which are set servo motors. Those servo motors can move the robot axes independently from one another and are commanded by the controller. In our case, the robot has got 6 axes.

For more details, refer to OM section 2.2 “Robot”.

## Jogging

Firstly, let's start with basic movements of the robot.
To jog the robot (move the robot), the automatic switch on the controller needs to be on T1 or T2 (T1 is the safer way to work with the robot, so we recommend to always use T1).
Then you need to enable the Teach Pendant with the Teach Pendant Enable switch.
You need to hold one of the Deadman switches on the teach pendant in the intermediate Position.
Finally you have to push a shift button and press one of the jogging buttons.
For more information refer to OM section 5.2.3 "Moving the robot by Jog Feed"
There are several different Cartesian systems that you can move your robot in.





For more details about the different coordinate systems, refer to OM section 3.9 "Setting Coordinate Systems".


For jogging you can switch between those different coordinate systems by pushing the "Coord" button.

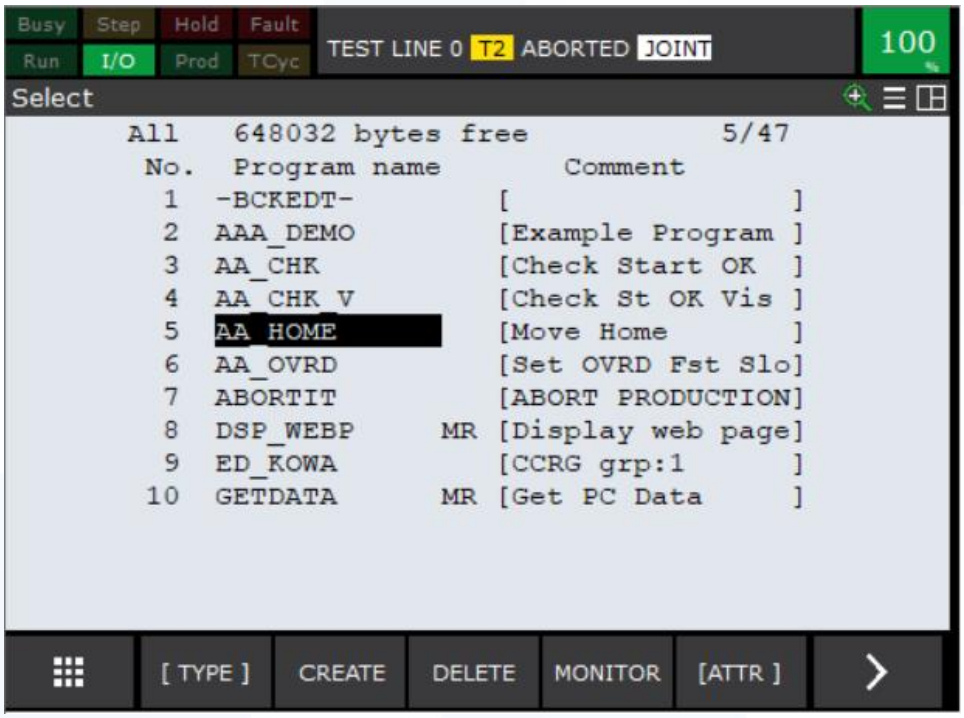
Here we changed the coordinate system from world to tool.

## Jogging


While you are in the Joint coordinate system, you can move joint by joint with the different jog buttons.(the joints are written on the jogging buttons)
In every other coordinate system you can move in the x-,y-, and z-axes by pushing the first three jogging buttons, and do rotations around those axes with the following three buttons.
Now try and jog the robot in the different coordinate systems and in different directions until you get used to move the robot. Pay attention to the differences in jogging in the different coordinate systems.

## Selecting a program

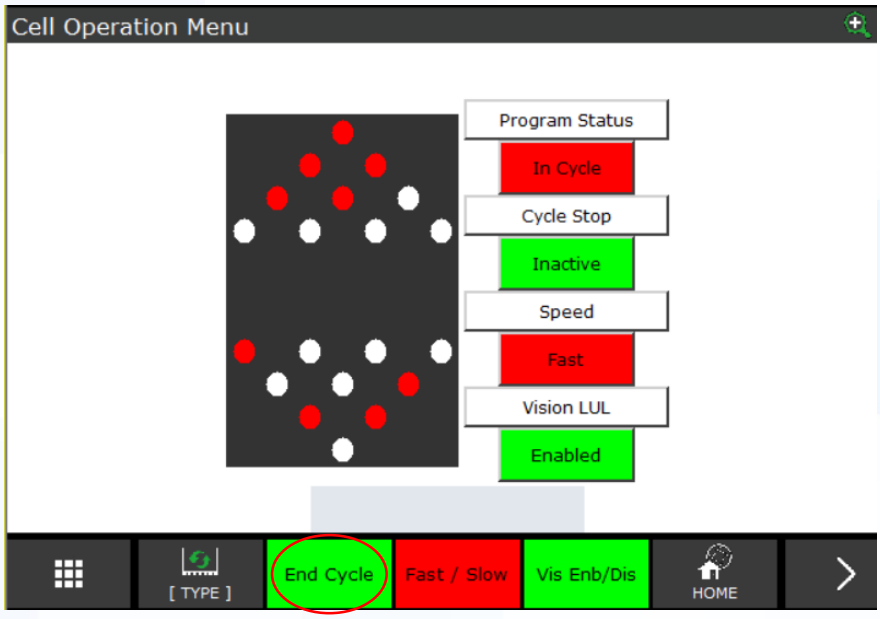
To select a program, press the “select” button.
Now you can choose a program in the list.


To enter or select the program press the “enter” button. You then get into the program and that program is then selected. The selected program is displayed in the top part of the Teach Pendant display.

## Starting a program

To start a program, you need to clear any obstacles in the motion range of the robot.
Then you need to close the fence (in our case the cell door).
You need to put the Three-Mode Switch into auto mode and disable the Teach Pendant switch.
Now the robot is in automatic mode.
If you press the cycle start button on the controller, the current program will start.
Now, select the program AAA_Demo (see previous page for how to select a program) and press the cycle start button.
The program asks you if all the pieces are in the right spot. Check whether they are and proceed (press F2).
To stop the program again, you can press F2 again to End the cycle after the end of the cycle.


Refer to the Education Cell Manual for more information.

## Recapitulation

After this exercise, you should know the basics of the robot and the robot controls.
You should be able to jog the robot in the different coordinate systems and also joint by joint.
You should be able to select a program and start it in the automatic mode.

FANUC

Exercise 2

# Simple TCP Teaching

FANUC

Exercise 2

# Simple TCP Teaching

## Table of contents

## Background

TCP stands for “Tool Centre Point”. This is a position defined relative to the end part of the robot (J6 faceplate). It is normally defined as the “Working Point” of the robot end-effector, for instance the centre of the gripper or the end of the arc-welding torch. It is important to set this position correctly both for teaching, since the jogging keys can be used to move the robot around the TCP, and for running, since the robot software controls the position and motion of the TCP.

A correct TCP setting is making sure that the coordinate system of the tool are along the tool axis. This allows to move the tool in a much more intuitive way while jogging. A badly taught TCP could mean that in operation the tip firstly would not be in the place that the operator and the robot would like it to be and secondly, rotating the tool tip will result in an unwanted movement of the tool tip while touching or even damaging objects or the robot itself. (normally the TCP shouldn't move while rotating around one or more axis)

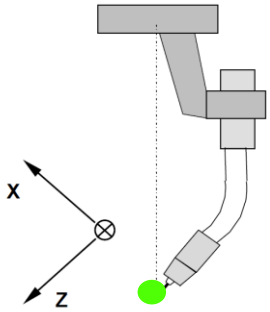
When the TCP is taught correctly, attitude change of the tool will not result in a change of speed or unwanted movement of the tool centre point, but the robot will adjust its movement so that the speed and movement of the tool will be correct.

For examples of TCPs see following page.

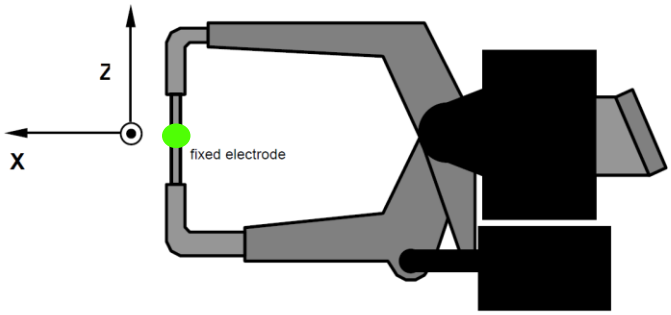
Examples of difference in movement with different TCP settings can be found on page 7.

## Background

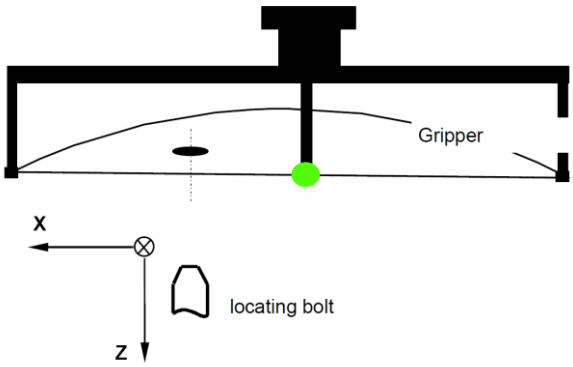
### Welding Torch



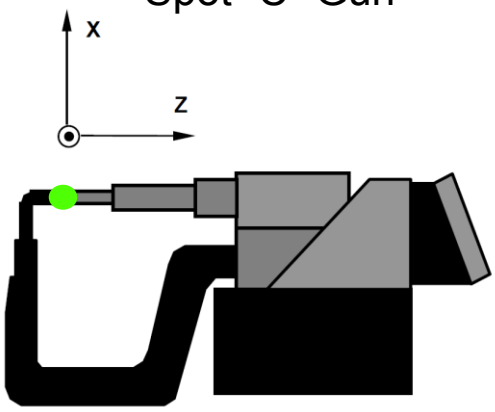

### Spot “X” Gun




### Gripper

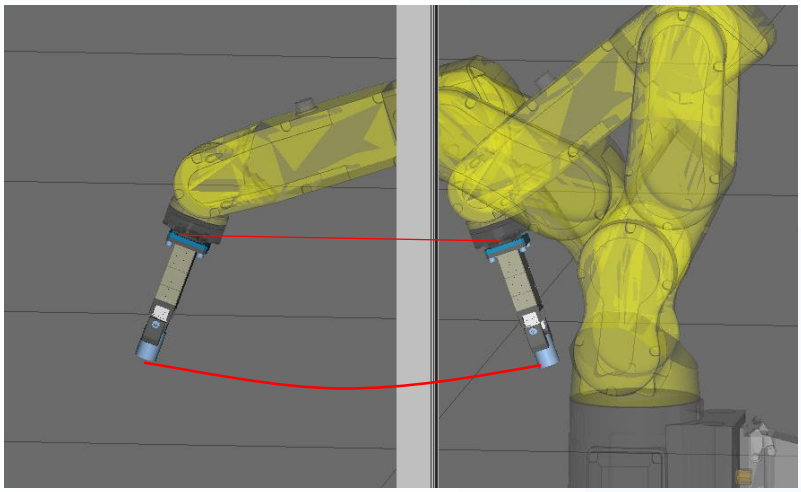



### Spot “C” Gun

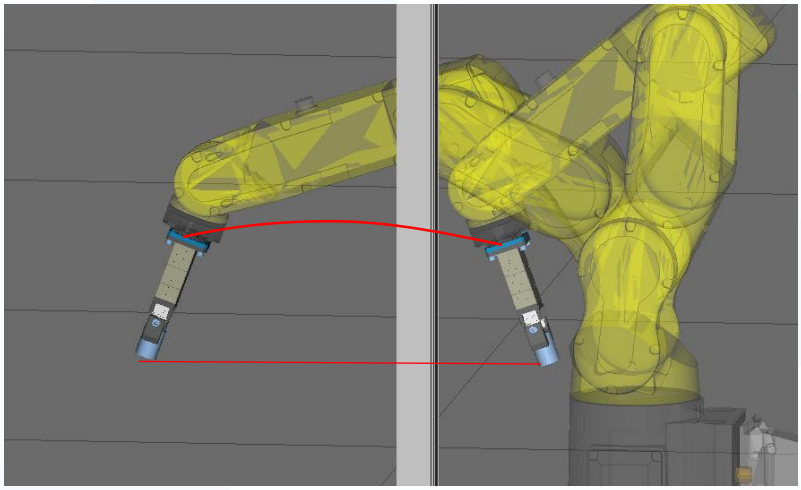



In these pictures, the coordinate systems represent the TCP coordinate system and the green dot represents the TCP.

## Background



In case the TCP is not set, default TCP is the flange of the robot and so while moving the robot and changing the attitude of the tool, the tool will not do a linear movement so this needs to be avoided.



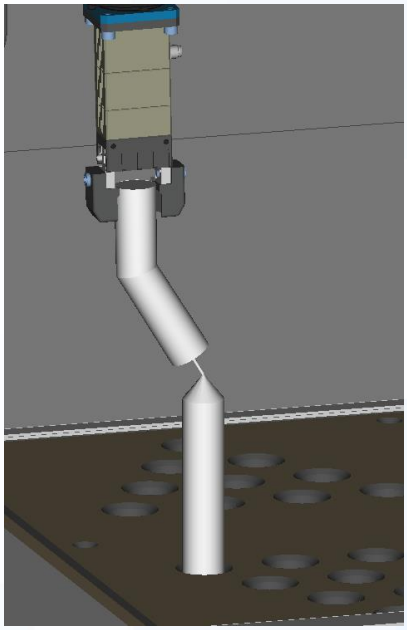
When the TCP is set according to the Tool, the robot makes sure that the tool moves right and adjusts the movement of the robot so that the motion is done according to the tool.

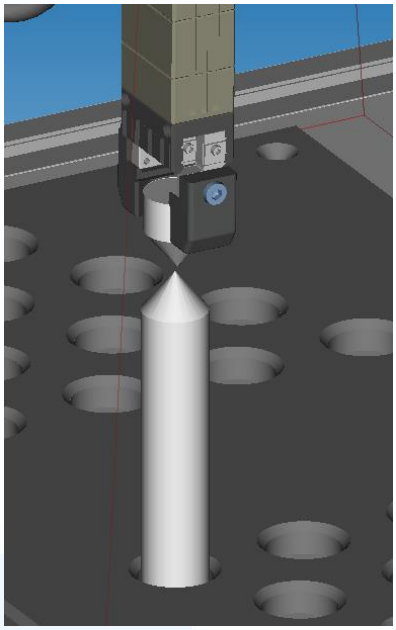
## Equipment

For this exercise you need:

- Education Cell
- Fixed Pin – on the table
- Movable Pins:
  - Simple TCP Pin in the gripper
  - Bent TCP Pin in the gripper (torch tool)

Description of the pins is in the Appendix.



## TCP Teaching Methods

There are several different ways to teach TCP:

- Three Point Method
- Six Point Method
- Direct List Method
- Two point + Z Method

For further details refer to OM section 3.9.1 “Setting a Tool Coordinate System”

## Three Point Method (TCP Auto Set)

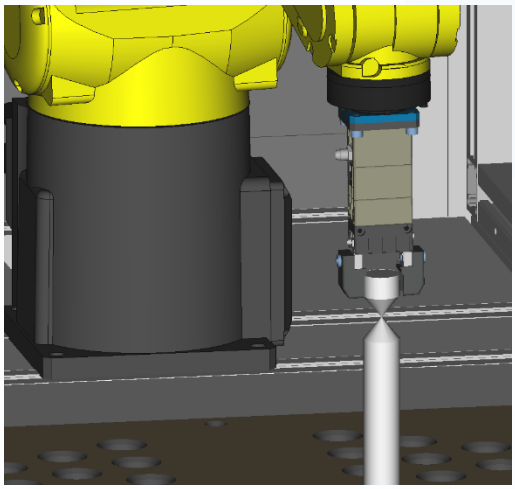
For this method use the simple TCP pin.

With this Method, the robot calculates the Tip Point by using three different approaches of the tip to an exact same point.
To do this approach the same point with your tip three times from different angles and directions and record them.
After you have set the TCP with the Three Point Method try and rotate the TCP using jog movement and observe if you can see any movement of the tip. If you do please reconsider the Manual and eventually do the Three Point Method again, until you are happy with the results.

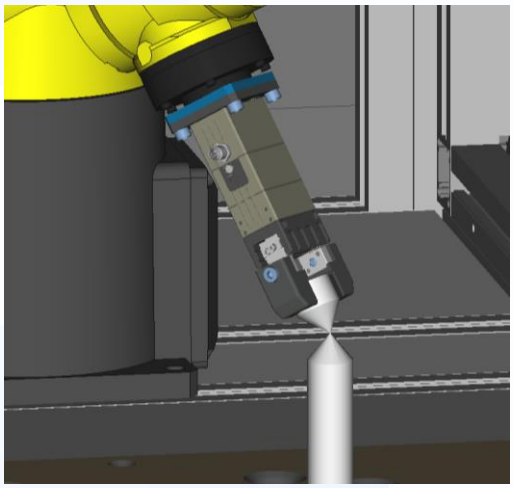
For further information refer to OM procedure 3-14 “TCP auto set (Three Point Method)”

Following are example positions that you can use, that will work, but you can also use your own positions. Just keep in mind that by taking more extreme positions, the calculations of the TCP will be more accurate.

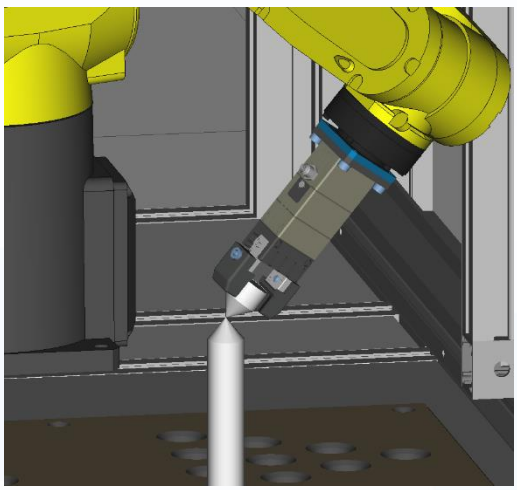
## Three Point Method (TCP Auto Set)



First Approach Point: from the top



Second Approach Point: from the right, near the robot with the tool tilted



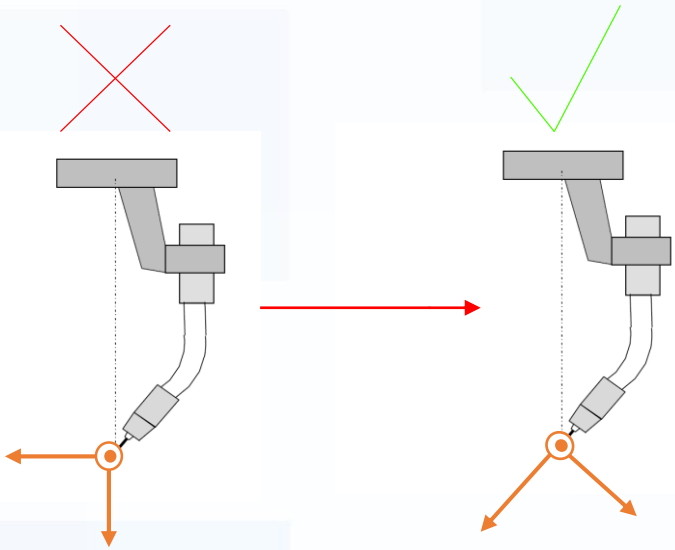
Third Approach Point: from the left, further from the robot and the tool tilted in the opposite direction

## Six Point Method

Why use Six Point method instead of Three Point method?

Three Point method is used for tools that are straight and in line with the J6 Faceplate. If this is not the case, by using the Three Point method, the tool's coordinate system is not correct because the axes aren't aligned with the tool tip.

By using the three point method, we will end up with the TCP in the left image. When using the six point method, we will be able to get the TCP in the right image.

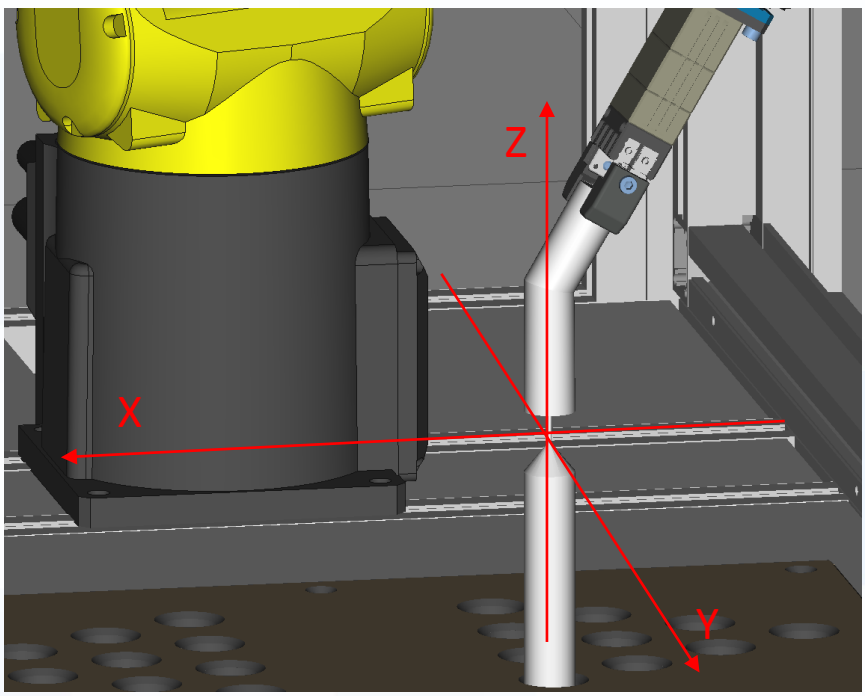


With the Six Point method, we are able to twist the coordinate system in such way that it is aligned with the tool tip, so the tool moves in a much more intuitive way (as seen above in the right image).

## Six Point Method

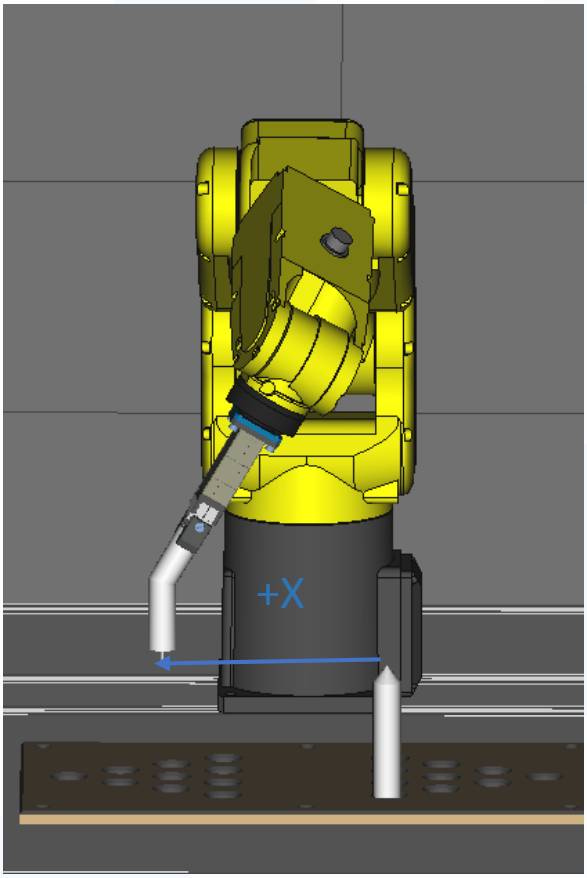
For this method use the bent TCP Pin.

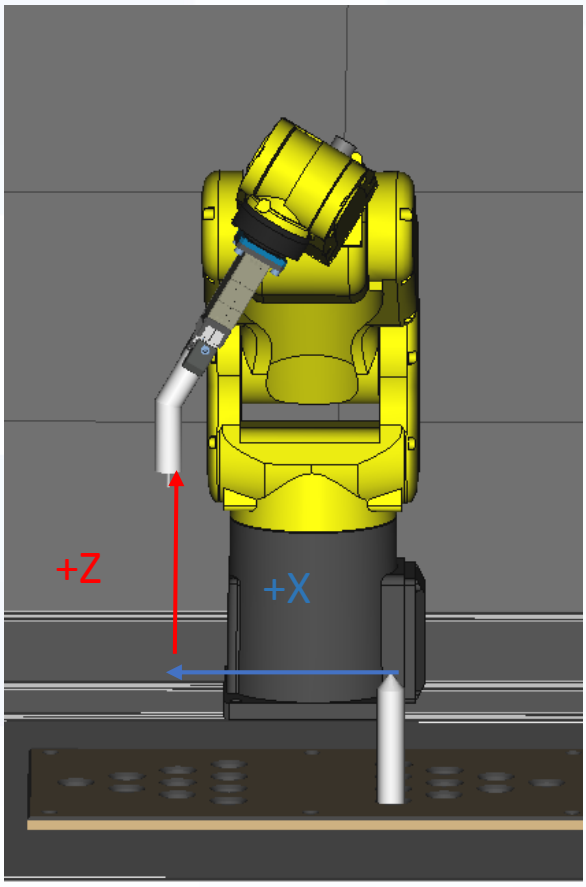
Firstly set the 3 approach points the same way as you did with the Three Point method. (You eventually set the tip point for the robot) Then put your tooltip vertically over the fixed pin tip and the bending of the tool should be in a 90 degrees angle to the X-axis of the World Coordinate System (WCS), then set Orient Origin Point. (Do this Operation as carefully and as precise as possible, as this is key to the precise coordinate system afterwards)



## Six Point Method

To set the coordinate system you need to move in the directions that you want your coordinate system to be. Because you oriented your tool bending in 90 degrees to the X-axis of the world coordinate system, by moving in Y-direction (in WCS) you can set the X-axis with a point on this line (this line will be your X-axis). Finally move the tool upwards (Z-axis in WCS) and set the Z-axis. Now your coordinate system is set according to your tool orientation.




Now the TCP should be set for the torch tool.
(Analog Procedure for the XY method)

## Six Point Method

To check if your TCP is set correctly check the following things:

- Check if while rotating, the tip is not moving.
- Check if moving in the different directions (X-,Y- and Z- in the tool coordinate system) that the tool is really moving in the expected way.

Finally, do the Three Point method again with the bend pin in the gripper. Watch the difference of moving the tool while jogging between setting the TCP with 3 point and 6 point method.

The bending of the bent TCP pin does not have to be in a 90 degree angle, but we strongly advise you to put it in line with one of the axes (X- or Y- in WCS) because this makes the definition of the axes of the tool much easier.

For further information refer to OM procedure 3-15 “Setting up Tool Frame Using the Six Point (XZ) Method”

## Direct List Method

For this method use the simple TCP Pin.

For the Direct List method, you need to put in all the coordinates and orientations manually. This method can be used if you know the exact dimensions and orientation of your tool. This is a much easier way to put in the information faster than with the Three Point or the Six Point method.
This method only works if we know the exact dimensions of our gripper and the piece.
As an exercise, go to the TCP set with the Three Point method. Change the coordinates with the Direct List method and watch the difference in behaviour while moving the tool in the Tool jogging mode. (For example, offset the tool Z-coordinates and watch the rotation around Y-axis with a new centre point in the point just set)

For further information refer to OM procedure 3-16 “Setting Up Tool Frame Using the Direct List Method”

## Two Points + Z Method

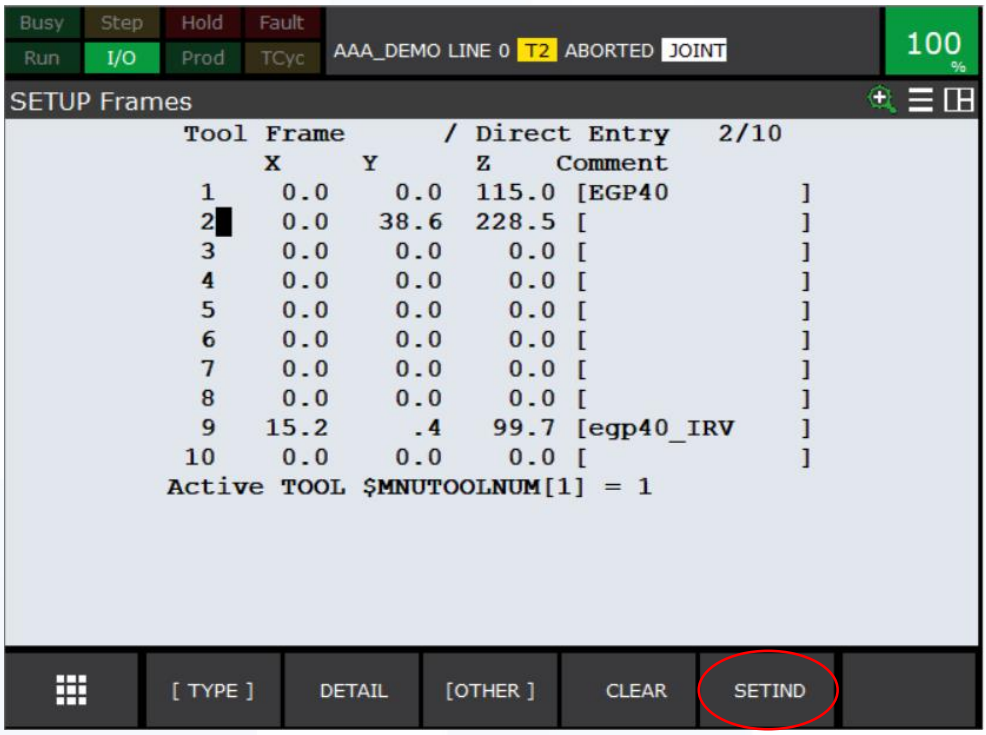
For this method use the simple TCP pin.

This method is a combination of the Direct List method and the Three Point method.
Firstly, input Z, W, P and R (by measuring those values)
Z is the measured distance between the J6 Faceplate and the tip of your tool (in our case the Simple TCP Pin). W, P and R are not important in our case because the Simple TCP Pin is straight and aligned with the tool (all angles are 0 degrees). If this is not the case you need to measure those angles and then put them in.
Then set the two approach points in the same way than with the Three Point method.

For further information refer to OM procedure 3-17 “Setting Up Tool Frame Using the Two Point + Z”

## Additional Information

Always make sure you set the right TCP setting with the SETIND function before you try and move your TCP because otherwise the old TCP setting is still active.

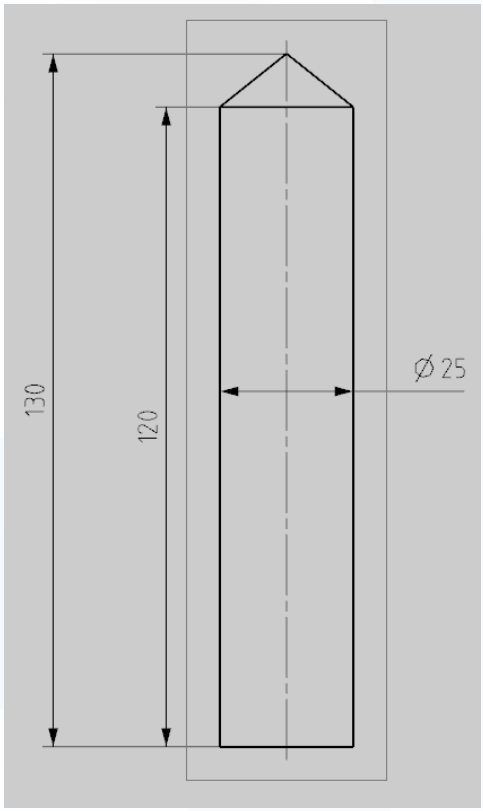


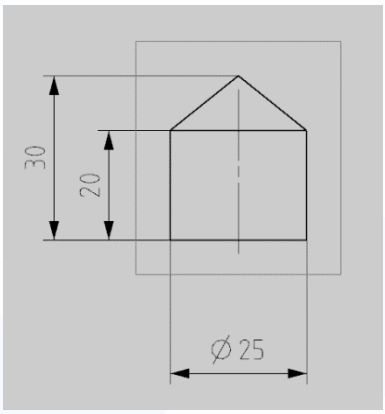
## Recapitulation

After this exercise, you should now be able to understand the concept and use of the TCP concept.
You should be able to set the TCP for the different types of tools with the different types of TCP setting methods and knowing the difference between those methods and the effects of those differences.

## Appendix

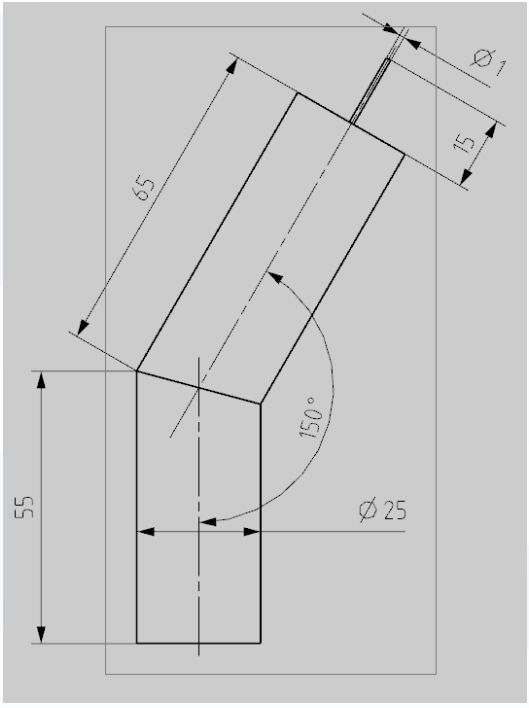
Simple TCP Pin – to be held in the gripper




Fixed Pin – to be fixed on the table

Bend TCP Pin – to be held in the gripper



FANUC

Exercise 3

# Simple Robot Programming

FANUC

Exercise 3

# Simple Robot Programming

## Table of contents

## Background

In industrial environment, we want our robot to do a certain work over and over again. In order to do this, we program our robot once so that it knows what to do and after this, the robot just keeps on doing the same work. This makes the work process extremely predictable and therefore highly efficient.

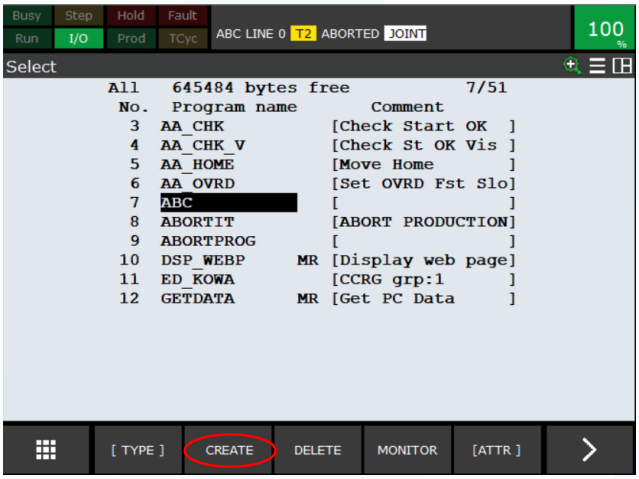
## Equipment

For this exercise you need:

Education Cell

## Creating a new Program

The first thing we need to do, is to set the Three-Mode Switch to T1.
Second, we need to enable the Teach Pendant
We then start by creating a new program.
To achieve this, push the “select” button. Then press F2 “Create”.
You then get asked a name for the program. Give the program a name and press Enter. You are now in the program and ready to input your first commands.




Press F2 (Create) to create new programs.

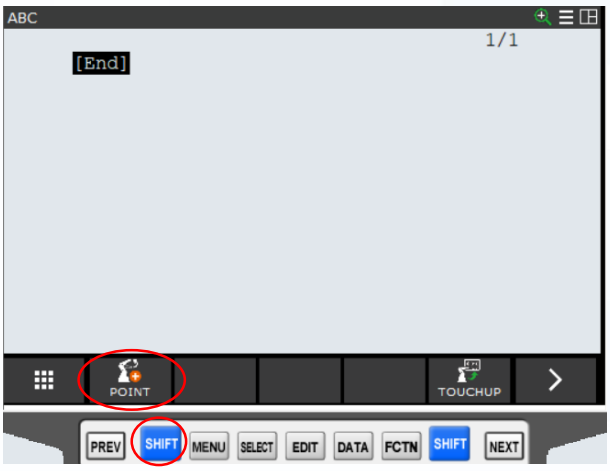

After you have created your new program, you are left with a blank program page.

For more information about creating a new program refer to OM procedure 5-2 “Registering a program”.

## Insert Position Data

Now jog the robot to a random position.
To record this position you have two possibilities:

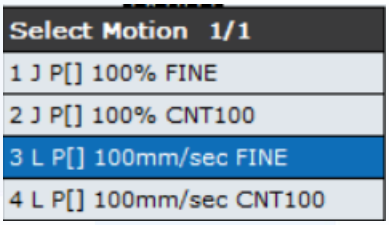
Press shift+Point




When doing this, joint motion will be set as default motion.

Press Point

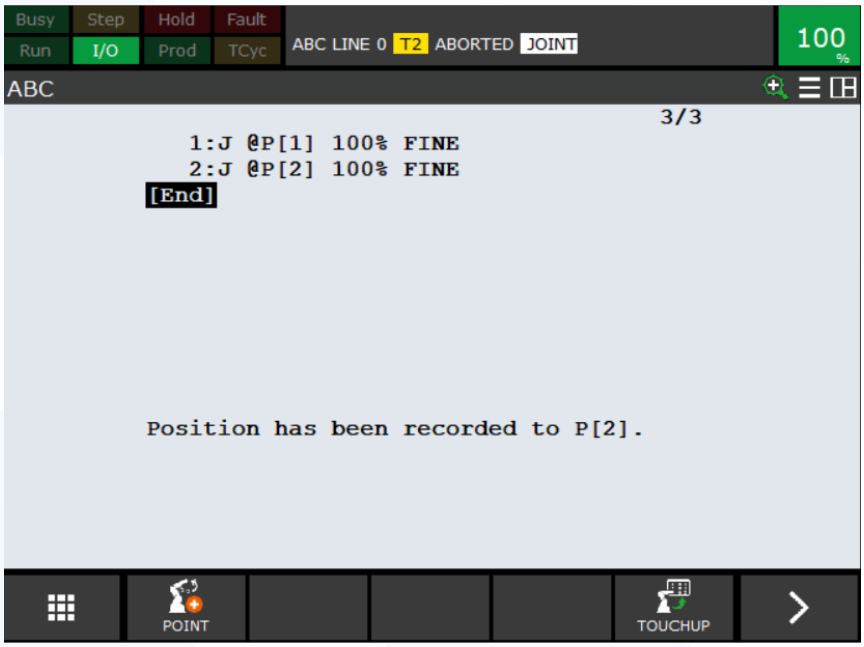

When doing this, you will be asked which motion to use.




For more information about different motion options refer to OM section 4.3.1 “Motion Format”, 4.3.3 “Feed Rate” and 4.3.4 “Positioning Path”.
For more information about teaching a new point refer to OM procedure 5-4 “Teaching a motion instruction”.

## Insert Position Data

Now, jog to a second point.
Do the same procedure again whilst using shift+point.
This is now a very simple and basic program, which can already be executed.



## Execute first Program

Next step is to execute your first program and observe if the program is doing what you want it to do. For this adjust the override to 10% with the override buttons. This is to make sure the robot does not operate at full speed, but at only 10%, so if something is not happening as expected, you have time to stop the robot before serious damage or injury might occur.

## Execute first Program

Then set the program to step. In order to achieve this, press the “Step” button.

For further information about step mode operation refer to OM section 6.3.3 “Step Test”.

In our case it is not very important as this program only has two steps, but it becomes much more important for bigger programs.
This setting provides, that the program only executes on step after another and only if you tell the program to continue.

## Execute first Program

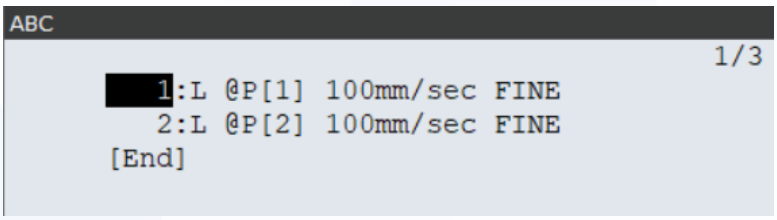
Now to execute the program, you need to keep the Deadman switch pushed, as well as the shift button and press the “FWD” (forward) button.
The program now executes the line currently selected.
After your program has successfully finished the program, you can gradually increase speed and if you feel confident that everything works, you can press the “step” button again and the program will be executed until the end.

**PLEASE NOTE THAT**

The Shift and the Deadman switch need to be pushed the whole time!

## Change of Motion

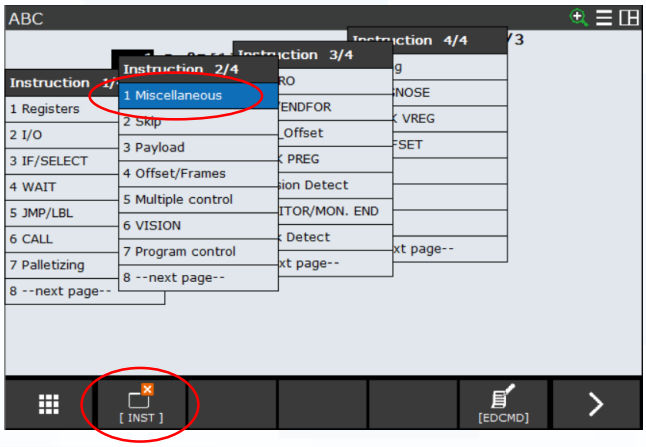
On default, the motion is set as joint motion. To change this, move with the cursor on the motion you want to change and press “Type”. You are then able to change between the 4 different motion types.
Set the two motion types to L (linear) and observe if you see any difference in the execution of the program.
You are also able to change the speed of the motion in the program. In order to do that, move the cursor to the speed setting. If you press the “Type” button you can input the type of speed.
Change the speed type now to mm/s and the speed to 100mm/s.
Execute the program again and observe if you can see any difference.

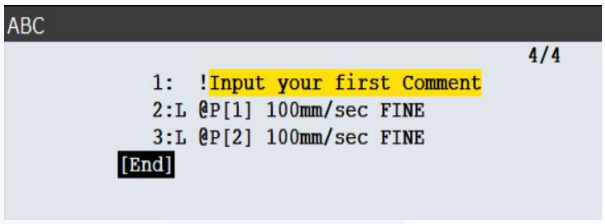



For more information about speed types and Feed rate refer to OM 4.3.3 “Feed Rate”.
For more information about changing motion and speed settings refer to OM procedure 5-3 “Changing a standard motion instruction”.

## Comments

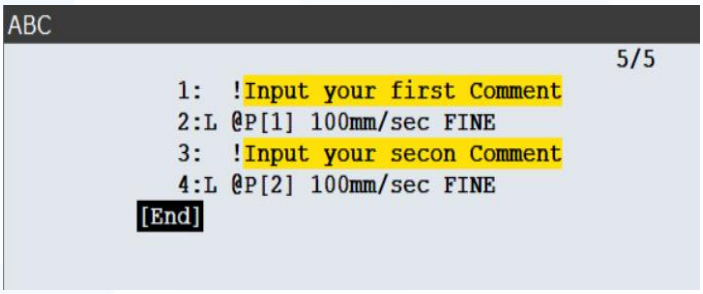
You have successfully created and executed your first program.
In order to allow other people to understand what you were doing in your program, we want you to put in some comments to explain the different operations and positions.
This is not very important for our program because it is only two lines, but later with programs bigger than 100 lines, it makes the work of understanding a program a whole lot easier.
To do this we firstly need to put in a new line in front of our command line. Move the cursor to the first line, press F6 (next) and then F5 (EDCMD) and press “Insert”. You will be asked how many lines to add. Add 1 line.
Then press F1 (INST), “Miscellaneous” on the second page and then “Remark”. After pressing “enter”, you can enter a comment and then afterwards, close the comment by pressing “enter” again.



## Comments

Put a second comment between the first and the second motion instruction.
You can also put comments directly into the position. In order to do this, move the cursor on the position you want to comment end press enter. Then you can enter your comment and press enter again.
Your program should now look like this:




You have now learned the basics of programming the robot. You can now put in more points and different motions and always put in comments.

## Warnings

Always jog the robot and execute the programs at low override settings. It may be that you programmed a wrong point, or you are not completely aware of the movement of the robot, so having the override at a low setting means that there is enough time to stop the robot before anything bad happens.

Until you are completely sure that your program works without any problems or bugs execute the robot only on step mode. This will prevent you from getting surprised and not being able to interact in case of an unwanted movement.

Unless you haven't tested your program on 100% override and without the step function, do not operate the program in auto mode because you are not completely sure that your program will work as desired.

When executing program on auto mode, execute the program several times on a lower override before raising it to 100%.

## Recapitulation

In this exercise you should have learnt the basics about programming a robot.
You should be able to program a simple motion command and change the motion type and speed.
You should be able to add lines and comments to your program in order to make it easier to understand the program.
You should be able to add comments to positions.

FANUC

Exercise 4

# User Frame Teaching

FANUC

Exercise 4

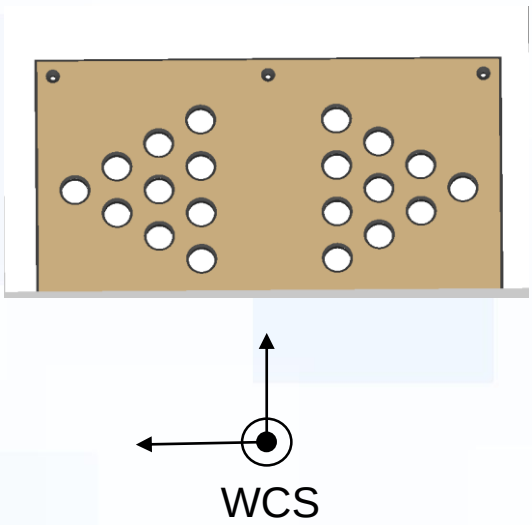
# User Frame Teaching

## Table of contents

## Background

The User Frame defines a coordinate system specific to the workspace, the robot is working on.
In our case this is not as important as in other cases because the solitaire board is in the same plane than the world and is perpendicular to the robot. Nevertheless moving the robot over the solitaire board in World Coordinate System (WCS) is not very practical, because the holes are not on an axis and so, moving from one hole to another is not so easy in WCS. That's why we define a User Frame that is different to the WCS and aligned with the holes.
This function is even more important if the robot is mounted in an inclined plane, or if the workspace is in the same plane but angled different from 0 or 90 degrees. Defining a User Frame in this case means moving the robot around is getting much easier and intuitive and teaching and jogging will become much faster.

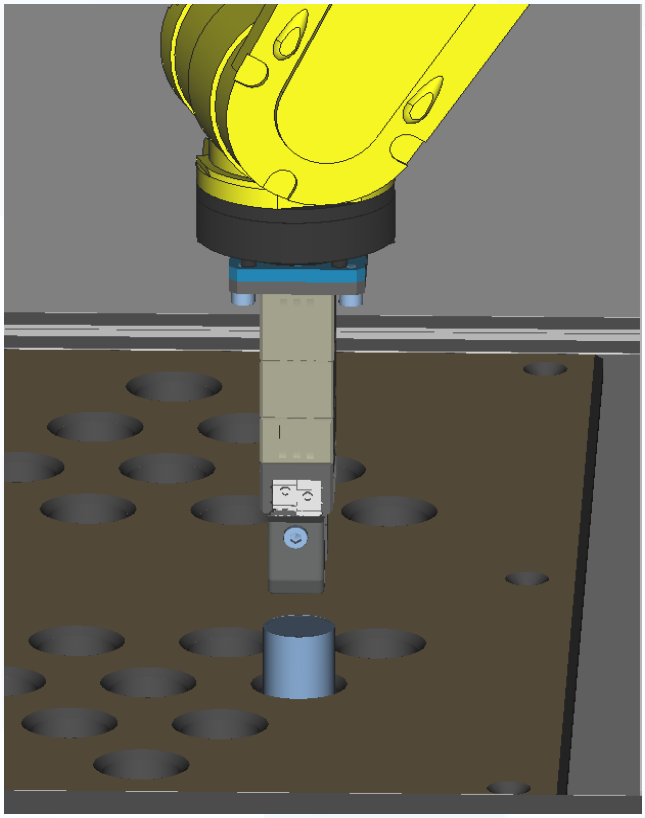
For moving on the solitaire Board the WCS (black CS) is not very practical



## Equipment

For this exercise you need:

- Education Cell
- Simple Cylindrical pin

## Defining a User Frame

First we start by setting up a User Frame on the left solitaire board side (as seen from the robot). We use the Three Point method. Use the User Frame 6.
We start by setting the origin. In our case we take the first hole as origin. This point is very important so do it carefully and with precision. While having the pin in the gripper, adjust the robot so that the pin is as exact as possible in the hole. Then set the height so that a piece of paper can just slip between the board an the pin.

**PLEASE NOTE THAT**

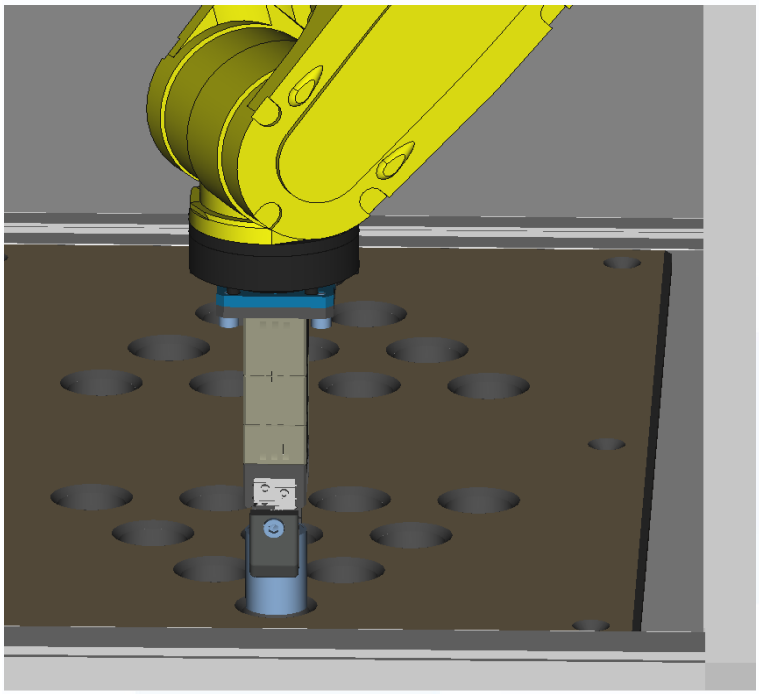
Z-origin is set on top of the board, not in the board, this is done because it is difficult to define the bottom of the hole with precision

A second point is needed to set the X-axis. So move the Pin to the nearest hole to the robot on that side of the board. Do the same procedure as with the first hole.
Lastly, move the part to a third point on the board. The position (X-, Y-) of this point is not that important, but keep in mind that with a bigger distance comes a greater precision so don't place it directly besides on of the other two points. The important part of this point is that the pin is at the same height than the other two points, so a piece of paper should just be able to slide under it.

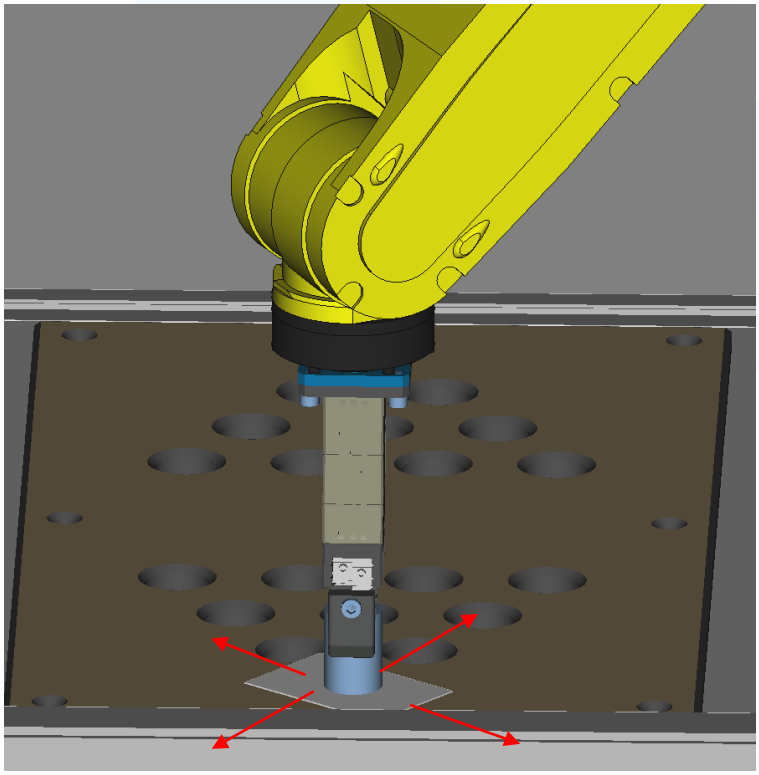
See following pages for images.

After you have set the UFrame press the SetInd (F5) button and set UFrame 6 as active User Frame.

## Defining a User Frame

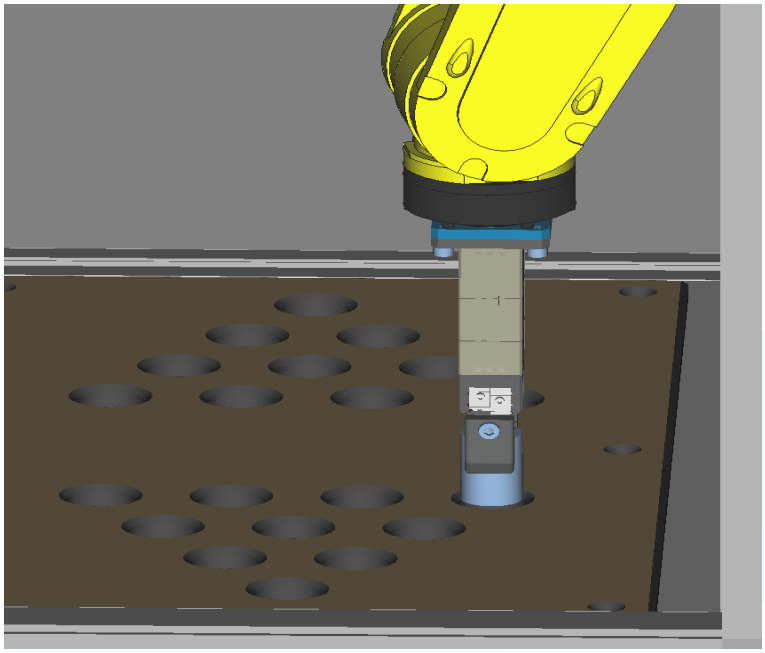


Origin set

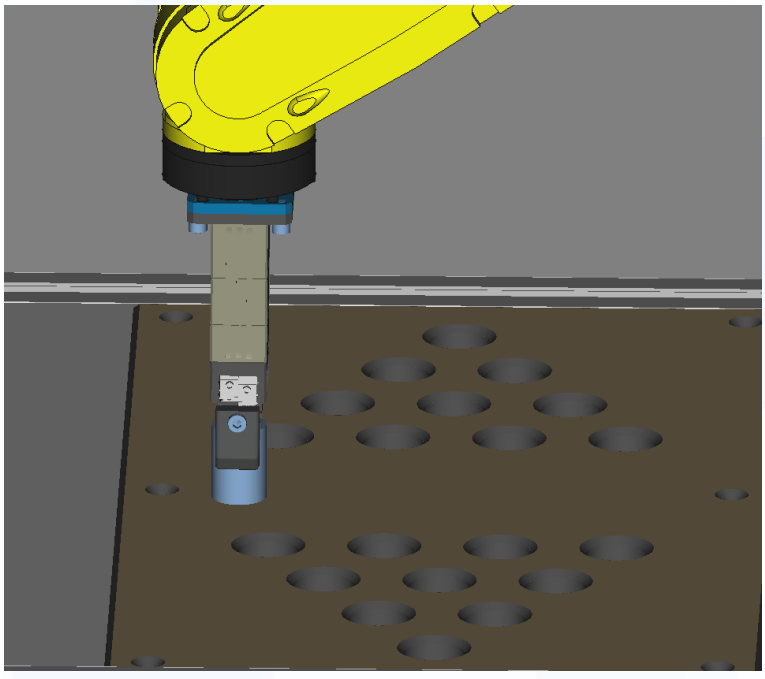


Make sure that a piece of paper can still glide between the pin and the board

## Defining a User Frame



X-axis set



Third Point set

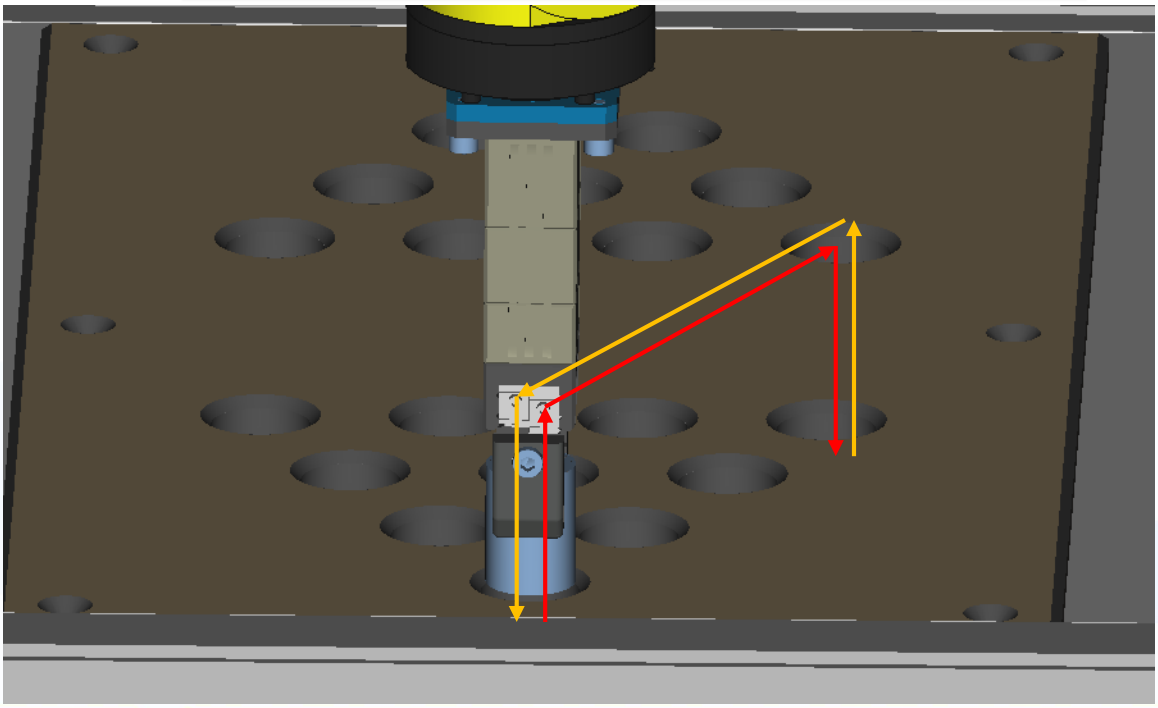
For further information about how to setup a User Frame refer to OM procedure 3-18 “Setting Up User Frame Using Three Point Method”

## First Program with a set User Frame

Now do a simple program while using the defined User Frame to move a pin from one hole to another.
Make sure, that you position the gripper with and without the pin exactly above the hole, a bit higher (some centimetres) before the gripper goes down, otherwise, the gripper will crash into the pin, or the pin with the gripper will crash into the board.
In the Appendix, we put an example program, so you can look how to do the program if you are not able to do one of your own.
Make sure to use the same UFrame whilst teaching points as whilst using those points in a program.

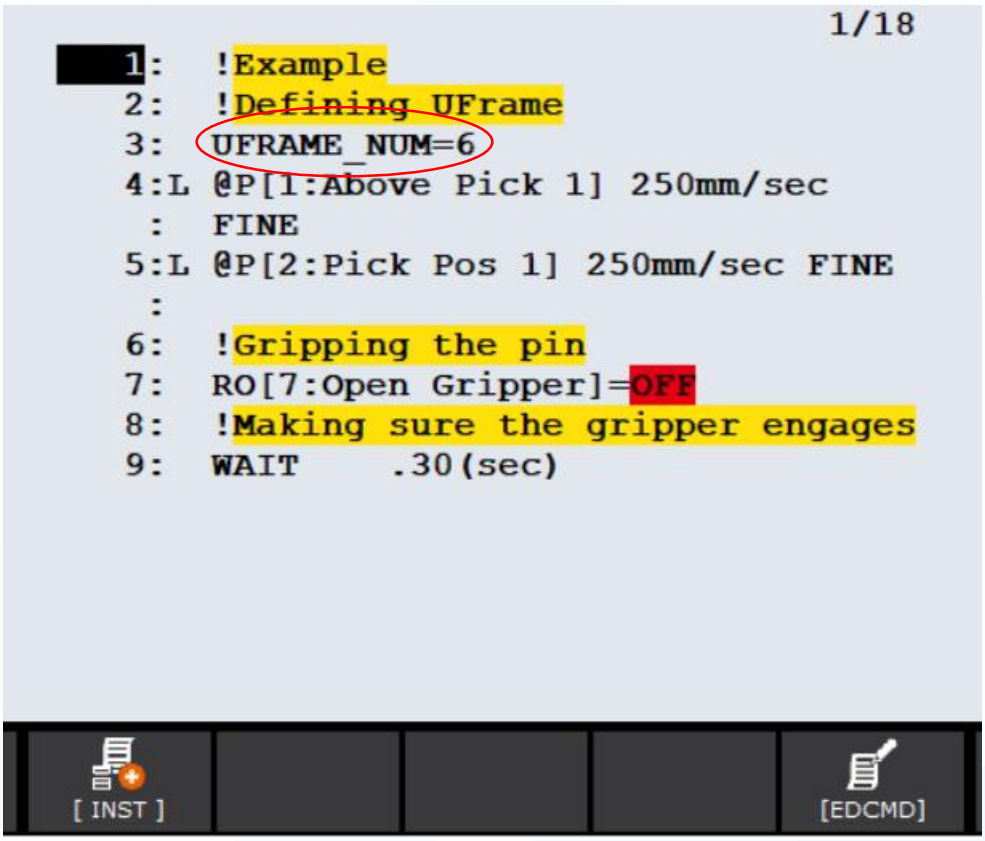
**PLEASE NOTE THAT**

If you are not able to do this program, we strongly recommend you to revise exercise 3



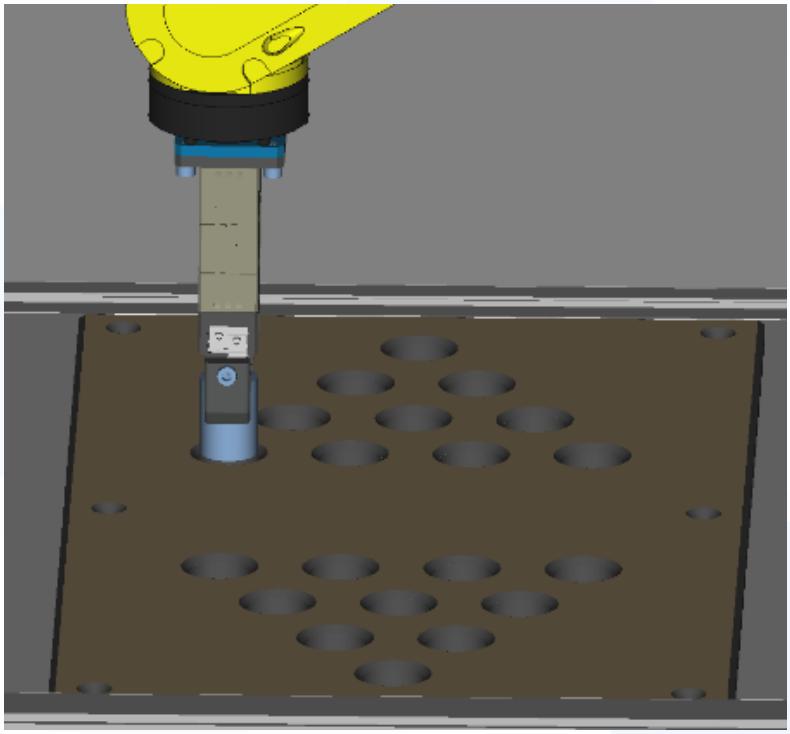
## First Program with a set User Frame

Make sure that you tell your program to use the right User Frame. Press F1 (INST), “Offset/Frames”, “UFRAME_NUM” and enter the UFrame you want to use. (In our case Frame 6)

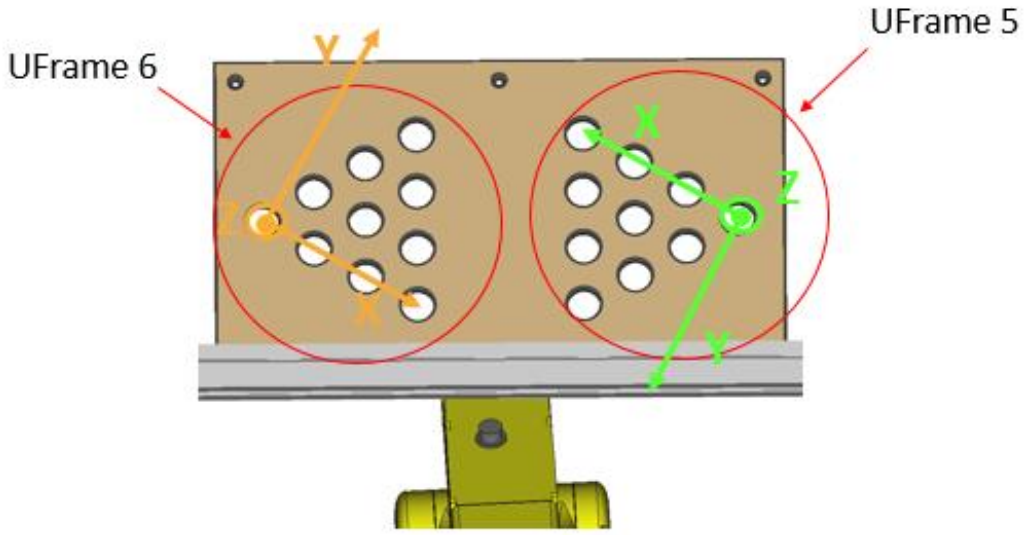


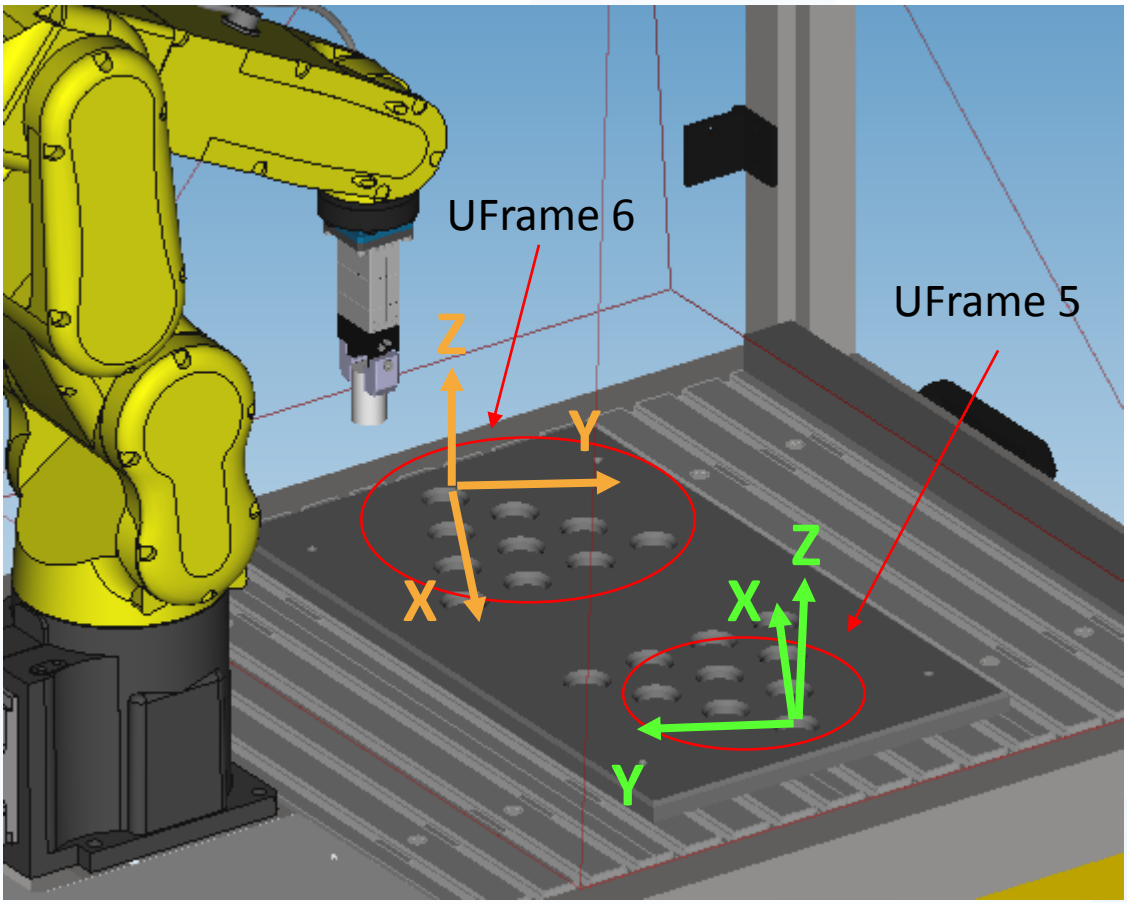
## Defining 2nd User Frame

If this program is set, the User Frame utilized is the User Frame of the left side of the Solitaire board. (UFrame 6)
We now want to use the same program on the other side
In order to do this, we set a new User Frame on the other side of the board, but this time using the hole the furthest away from the robot as X-axis point. Use User Frame 5.



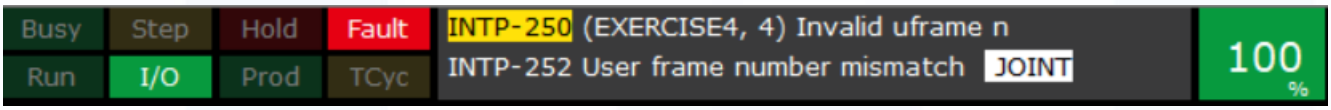
## Defining 2nd User Frame



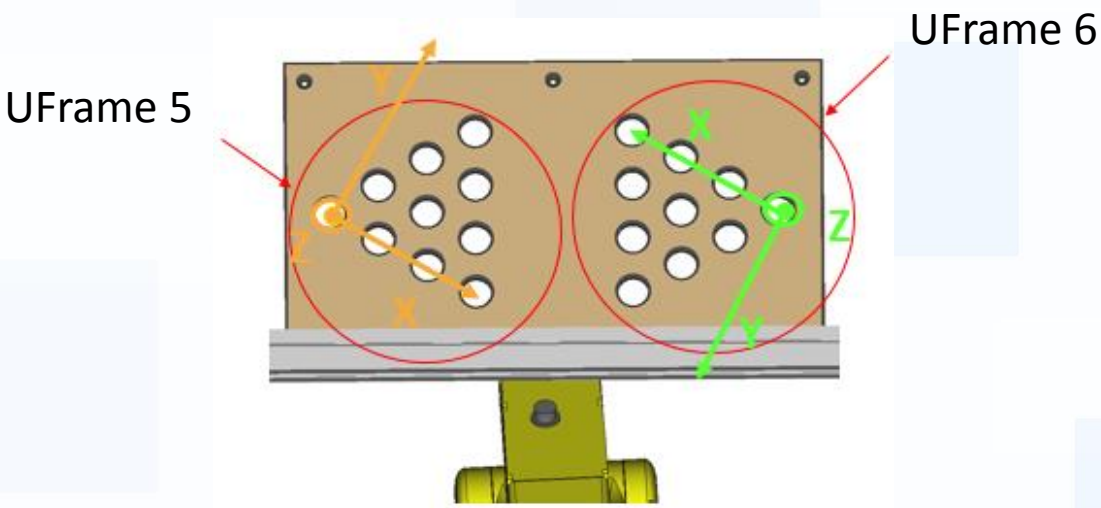



## Adapting Program to 2nd User Frame

Now try and use the same program again. Pay attention to change the User Frame in the program, because otherwise the program will do exactly the same as last time. (From UFrame 6 to UFrame 5)
You will notice that the program won't start and on the Teach Pendant appears a fault.

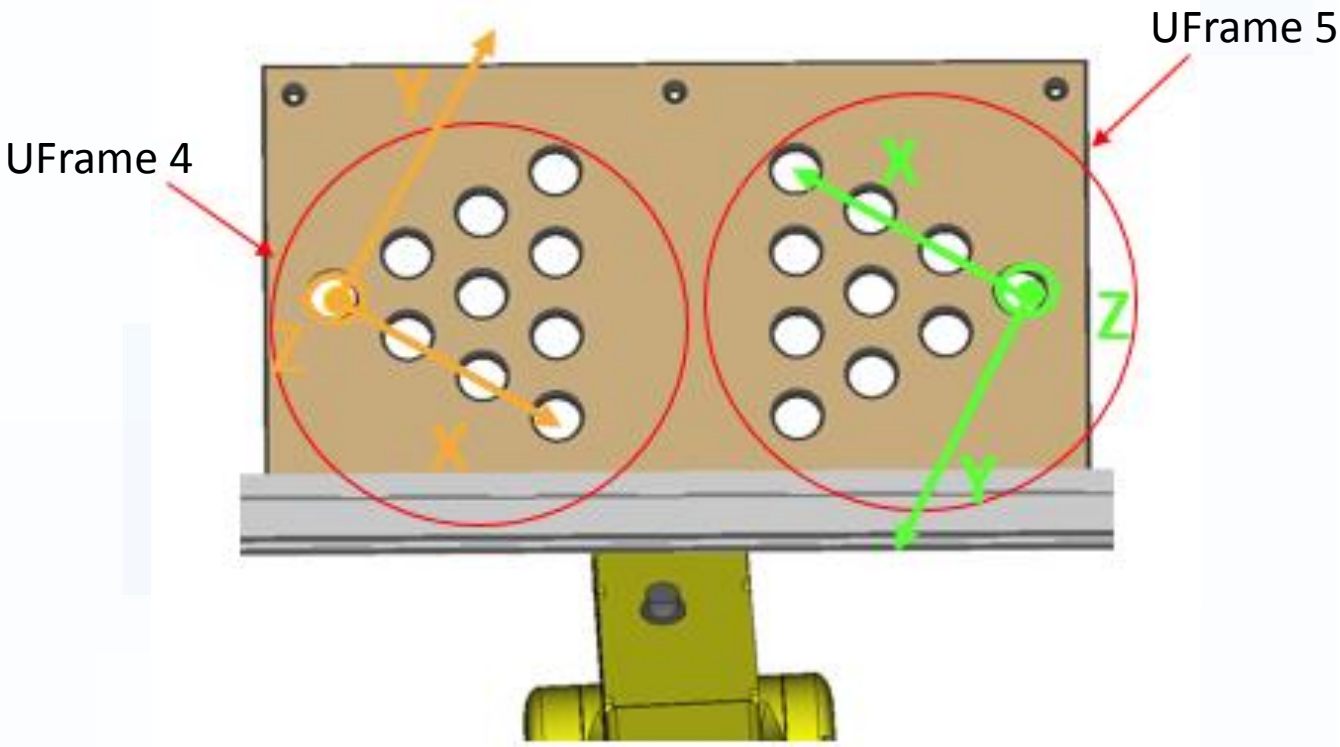



To make the program work, make the UFrame 6 the frame of the right side of the board.
Again change your program to UFrame 6 and now your program should also work on the right side.
Now we see that with this method, we can actually transfer a program from one side of the board to another by only changing the User Frame. But it still is unpractical, because now we have lost the User Frame of the left side, and to transfer the program to the left side again it would need the same work again, which is annoying.



## Better Programing Method

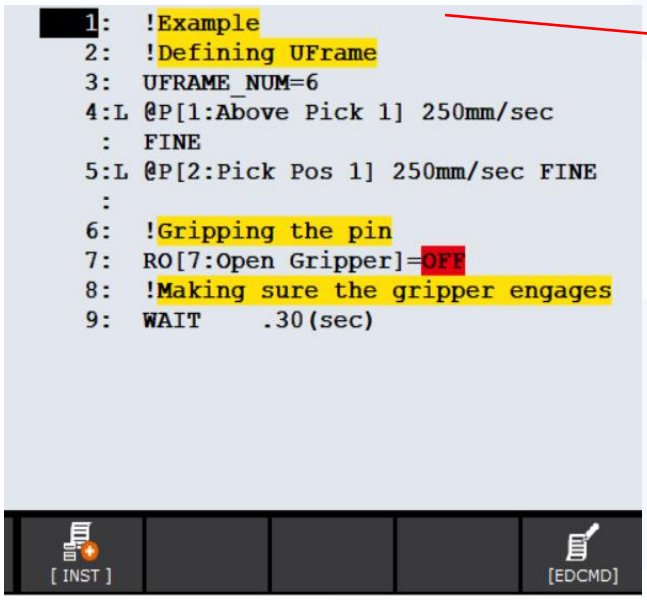
So now we want to try and create a program that can be used for both sides without the need of changing the User Frame every time. To do this we first have to program the two User frames again. So now program the left User Frame as UFrame 4 and the right one as UFrame 5. Clear the UFrame 6.

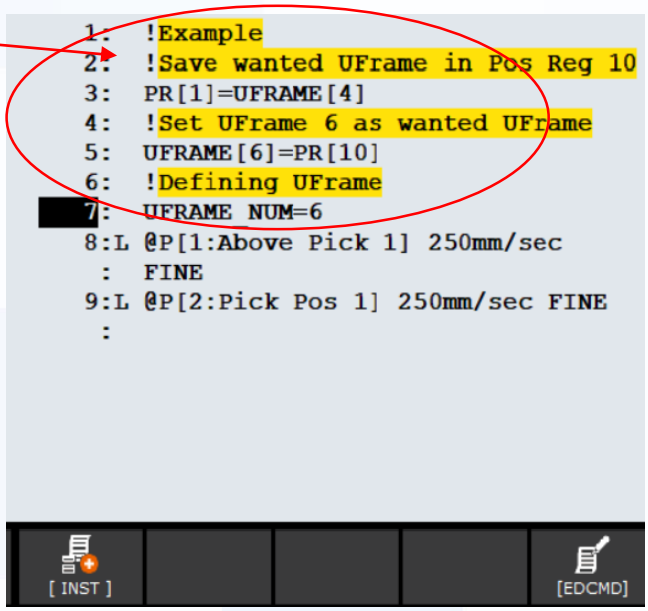


## Better Programing Method

Put in front of your program a position record order to record the position of the UFrame that you want to use. Then you set the UFrame that your program uses to the position record you just did. Now by only changing the UFrame in the third line, your program can work with both UFrames without having to change the UFrames every time.

For further information about position record register or register in general refer to OM section 4.5 “Register Instructions”.



## Warnings

**Attention!**

By changing the UFrame to the opposite side, it may be possible that the robot moves, or behaves in a way that you would not expect. This is due to the change of the Frame and we easily make faults while trying to understand the difference in movement. This means that you should use a slow override and always follow the exact path of the robot and check that there won't be any problems with the cable of the gripper. The first time the robot should be in step mode so that you can easily abort the program if you see any problems. When the robot does the program successfully without any trouble, you can gradually increase the override and later go to automatic.

If you do not pay attention, the robot might collide with the robot cell, or the cable might be torn apart which could result in potential damage of the robot and the cell.

## Recapitulation

Now you should be able to know the benefits of a UFrame in comparison to the fixed World coordinate system.
You should be able to set the UFrame with the three point method.
You should be able to transfer a program from one UFrame to another.
You should be able to change a program to work with more than just one defined UFrame and so make the program more flexible.

## Appendix

Example program to put the pins from one hole to the other.

**Attention!**

You need to teach the positions by your own, it is not enough to just copy this program as P[1] to P[6] were taught beforehand.

Remark:

Picking and releasing positions are not the same. This is due to the fact that the pin might not be entirely in the gripper, so descending with the pin in the gripper may result in a crash with the table, so it is better to set the releasing altitude a bit higher than the actual picking altitude

```
                        PAUSED      1/29
   1:  !Example
   2:  !Defining UFrame
   3:  UFRAME_NUM=6
   4:L @P[1:Above Pick 1] 250mm/sec
    :  FINE
   5:L @P[2:Pick Pos 1] 250mm/sec FINE
    :
   6:  !Gripping the pin
   7:  RO[7:Open Gripper]=OFF
   8:  !Making sure the gripper engages
   9:  WAIT     .30(sec)
```

```
                        PAUSED     16/29
  10:L @P[1:Above Pick 1] 250mm/sec
     :  FINE
  11:L @P[3:Above Pick 2] 250mm/sec
     :  FINE
  12:L @P[4:Release Pos 2] 250mm/sec
     :  FINE
  13:  !Releasing the Pin
  14:  RO[7:Open Gripper]=ON
  15:L @P[3:Above Pick 2] 250mm/sec
     :  FINE
  16:  !Calling predefined program
```

```
                        PAUSED     23/29
  17:  CALL AA_HOME
  18:  WAIT   2.00(sec)
  19:L @P[3:Above Pick 2] 250mm/sec
     :  FINE
  20:L @P[5:Pick Pos 2] 250mm/sec FINE
     :
  21:  RO[7:Open Gripper]=OFF
  22:  WAIT R[...]<>...
  23:L @P[3:Above Pick 2] 250mm/sec
     :  FINE
  24:L @P[1:Above Pick 1] 250mm/sec
```

```
                        PAUSED     29/29
  23:L @P[3:Above Pick 2] 250mm/sec
     :  FINE
  24:L @P[1:Above Pick 1] 250mm/sec
     :  FINE
  25:L @P[6:Release Pos 1] 250mm/sec
     :  FINE
  26:  RO[7:Open Gripper]=ON
  27:L @P[1:Above Pick 1] 250mm/sec
     :  FINE
  28:  CALL AA_HOME
[End]
```

FANUC

Exercise 5

# Offset and Position Register

FANUC

Exercise 5

# Offset and Position Register

## Table of contents

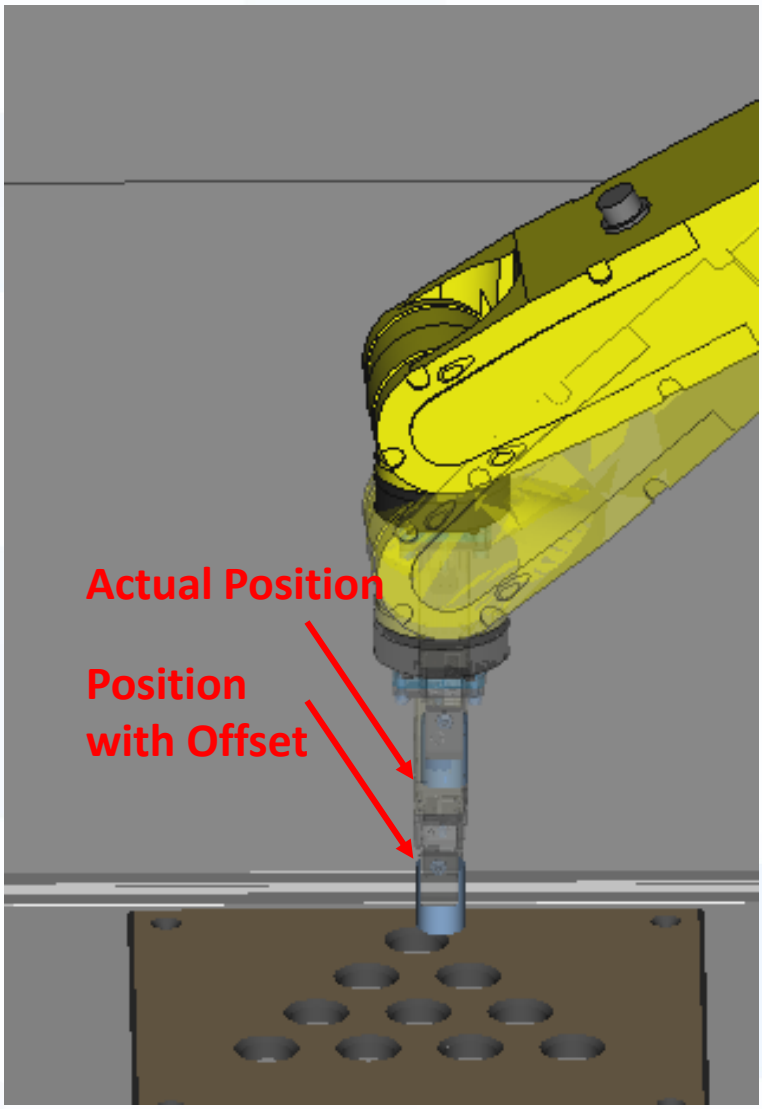
## Background

This exercise will be similar to the last one and most importantly, the effect/execution of the program will be the same (if correctly done).
With Tool Offset, you can as the name suggests it, offset your tool by a certain value. This can be particularly practical if you use the same position at different heights for example, as we do to move over a hole, pick up the pin and release this same pin.
Position Register is used to save a Position or an offset as a variable in the robot. The Position Register can be set before the execution of a program or while the program is executed.
When using offset and Position Register, we can make our programs much simpler and with less position teaching than before.

Reminder: to get further information and details about Position Register refer OM section 4.5 “Register Instructions”.

## Background

Here is a little example of what is meant by tool offset:



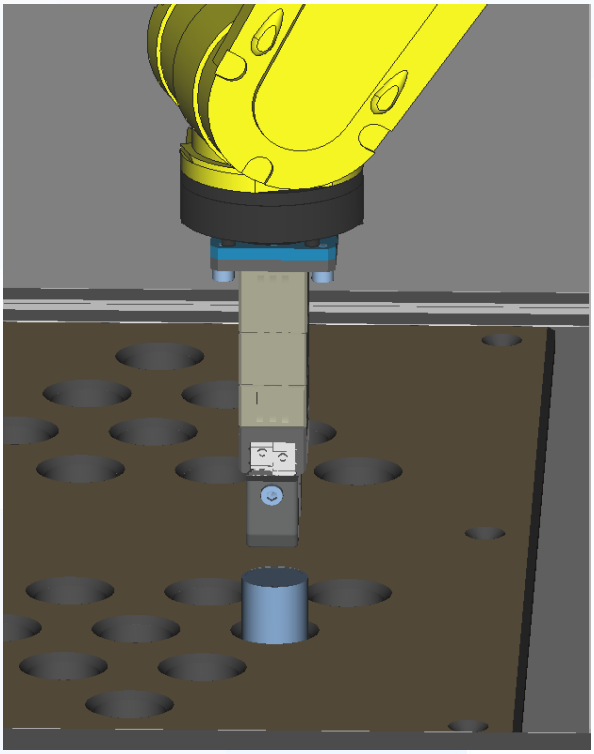

In this case, we have used a position and an offset in Z-direction. When adding this offset to the position, the Z-coordinate of the position will change and therefore the height in the World Coordinate System.

## Equipment

For this exercise you need:

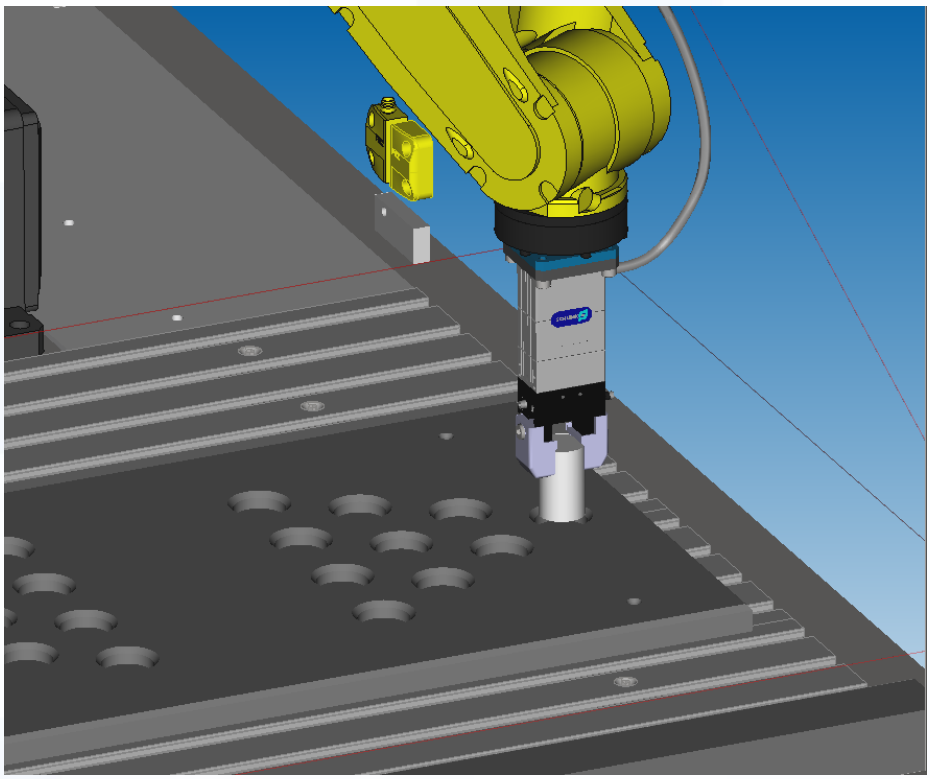
Education Cell

Simple cylindrical pin

## Input Position in Position Register

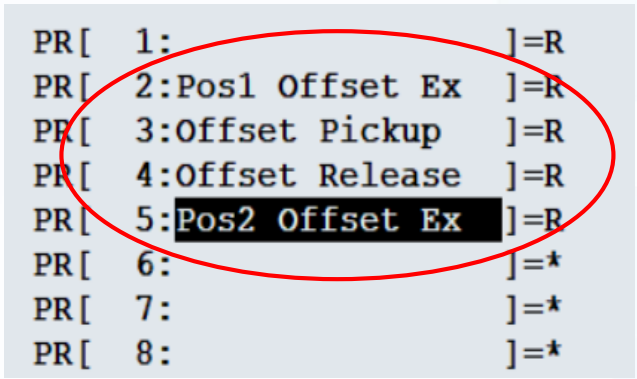
As already mentioned, Position Register is a sort of variable memory for positions and offsets.
To set a position, you can either input the values directly (with “Position” key), or you can record the positions from the robot, which we recommend you to do.(with “Record” key)
So firstly, start by setting the two positions you want to move the pins (you can also record more than just two positions, so you can put your pins in more than just two holes, but we only cover the two first holes, the other ones work exactly the same).

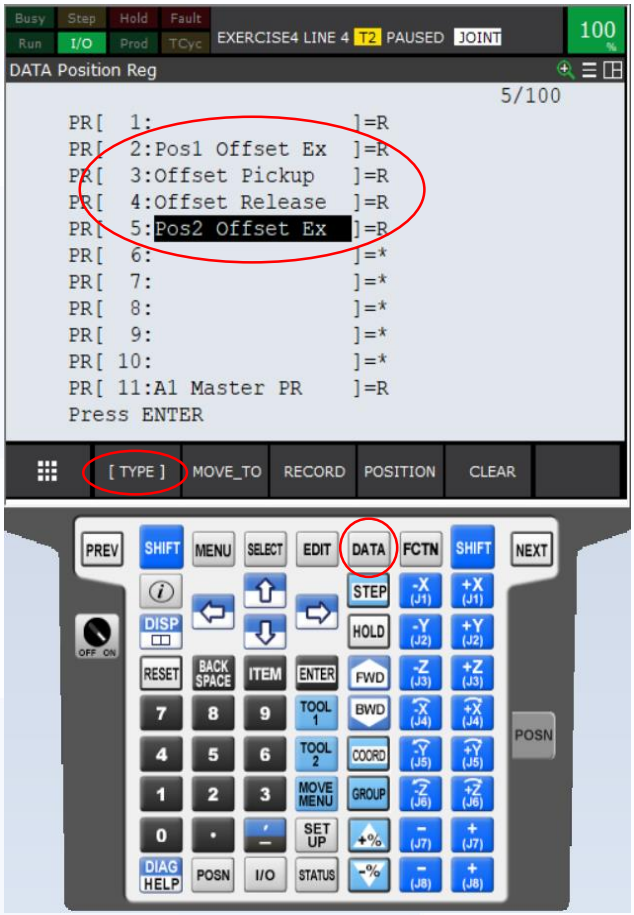


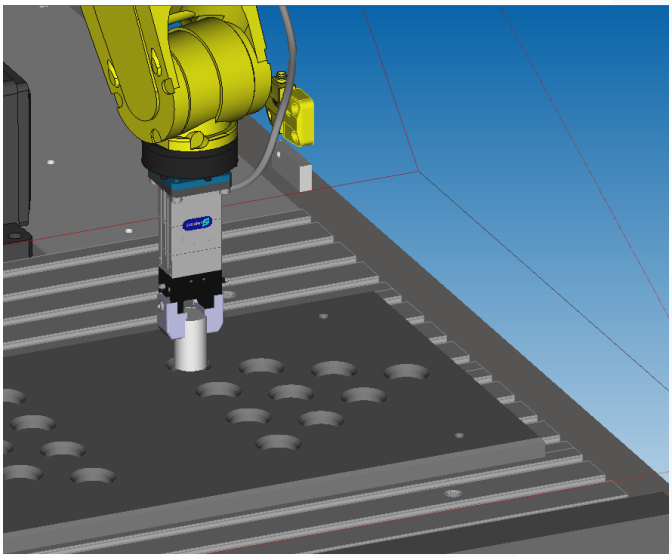
Again, put the pin exactly in the hole, and then go up on the z-axis to have the pin perfectly aligned with the hole and at the right height.

## Input Position in Position Register

To record the position into PR press Data, set Position Reg in [Type] and select the PR you want to record your position.



Then do the same procedure on the second hole you want to set. After you have set this position, go to position 1 and write down the Z-height of that position and input this same value manually (Position) into position 2 so that both positions have the same Z-value and can use the same offsets.

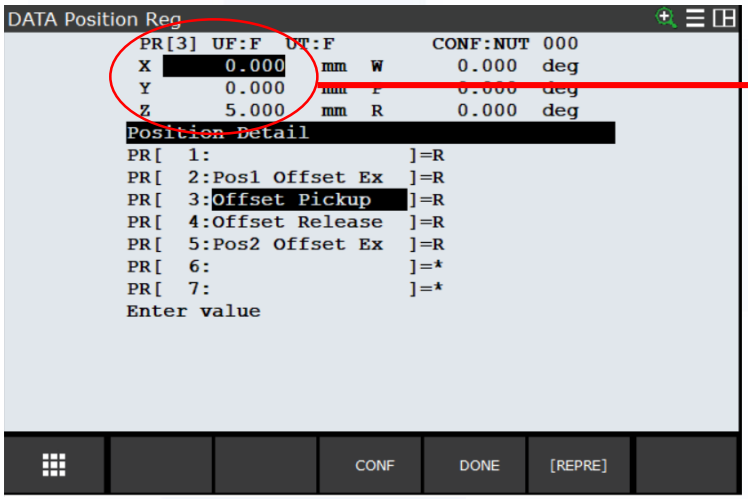
## Input Offset in Position Register

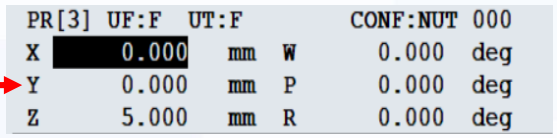
Now set an offset in the PR. The offset should only be in Z-axis and is depending on the position you chose as first position (position above the first hole).
This setting is based on trial and error. First put in a small offset and then gradually increase the offset until you get the offset you want to have. Pay close attention as to have the override at a small value so that you can stop the robot if it is moving to far or in the wrong direction.
The best way of doing this is by setting up a similar program than in exercise 4, while using the PR values set as positions and then using the offset. Then let the robot execute the offset an check whether the offset is right or not. If it isn't, go back to the PR and change your offset value slightly and try again.
We recommend also that you input two offsets: one offset to pick up the pin, which should be precisely the height at which the gripper can grip the pin, and a second one a bit higher to release the pin, in case the pin is not entirely in the gripper.

To get further information about how to program with offsets refer to OM section 4.11 "Offset Condition Instruction" and 4.12 "Tool Offset Condition Instruction"

## Input Offset in Position Register

To put offset into PR you operate in the same way as for the positions, only that instead of recording you go into “Position” and put in the value manually. Put every value except Z on 0 an then start with small values for Z (5 for example).



## Example Program

Now as you have set both Position and Offsets in the PR, you can create a new program, only this time instead of using positions, you directly use the positions set in PR and then with the tool offset function, you go down to grab or release the pin.

**PLEASE NOTE THAT**

As you have set the program in exercise 4 we assume you know how to do this program. If you don't, please go back and consider exercise 4. To make sure your program is right we put a reference program in the appendix.

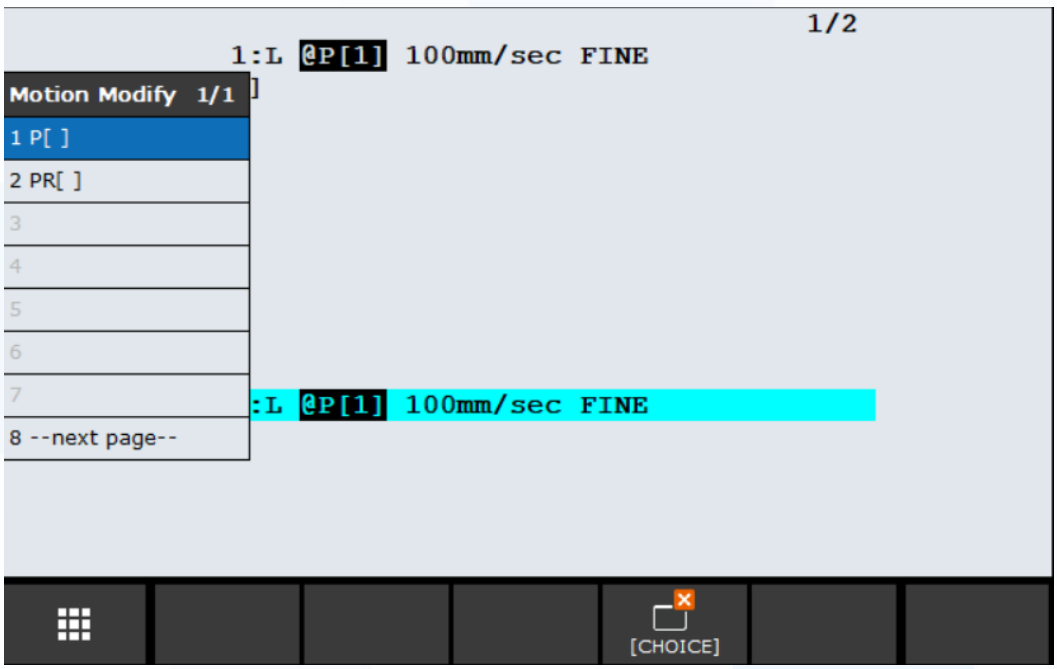
To use a position from PR, simply press the record key. Move with the cursor to the position and change the "Type" to PR then set the PR value you want to use.
As you have set the program you will probably notice that it takes less time to program this way than it took with the other (simpler) method.
You need however to pay attention, that this method is depending on which UFrame and which Tool Frame you use. The use of a wrong frame will result in an unwanted movement of the robot, which could result in human injury and/or damage of the robot.

## Example Program

First record a random position then move with the cursor on that position

```
                                                    1/2
   1:L @P[1] 100mm/sec FINE
[End]
```

You then press the “Choice” button and select PR[ ] and set the wanted value.



## Recapitulation

During this exercise you should have learned the basics about inputting positions and offsets in the position register.
You should be able to record and set certain values in the PR.
You should be able to use those values in your program.
You should be able to do a program entirely based on PR without the need of recording every single point.

## Appendix

Here is the program that we used for this exercise. Please note that this is according to our values saved in our PR, so those values can change according to your own settings.

```
                                                    1/34
   1:  !Move to Pos1
   2:L @PR[2:Pos1 Offset Ex] 100mm/sec
    :  FINE
   3:  !Make the pickup offset active
   4:  TOOL_OFFSET CONDITION
    :  PR[3:Offset Pickup]
   5:  !Move again to Pos1 with offset
   6:L @PR[2:Pos1 Offset Ex] 100mm/sec
    :  FINE Tool_Offset
   7:  !Grab the pin
   8:  RO[7:Open Gripper]=OFF
```

```
                                                   29/34
  24:  TOOL_OFFSET CONDITION
    :  PR[3:Offset Pickup]
  25:L @PR[5:Pos2 Offset Ex] 100mm/sec
    :  FINE Tool_Offset
  26:  RO[7:Open Gripper]=OFF
  27:L @PR[5:Pos2 Offset Ex] 100mm/sec
    :  FINE
  28:L @PR[2:Pos1 Offset Ex] 100mm/sec
    :  FINE
  29:  TOOL_OFFSET CONDITION
    :  PR[4:Offset Release]
```

```
                                                   15/34
   9:  !Move to Pos1 without offset
  10:L @PR[2:Pos1 Offset Ex] 100mm/sec
    :  FINE
  11:  !Move to Pos2
  12:L @PR[5:Pos2 Offset Ex] 100mm/sec
    :  FINE
  13:  !Make the release offset active
  14:  TOOL_OFFSET CONDITION
    :  PR[4:Offset Release]
  15:  !Move to Pos2 with offset
  16:L @PR[5:Pos2 Offset Ex] 100mm/sec
```

```
                                                   34/34
  28:L @PR[2:Pos1 Offset Ex] 100mm/sec
    :  FINE
  29:  TOOL_OFFSET CONDITION
    :  PR[4:Offset Release]
  30:L @PR[2:Pos1 Offset Ex] 100mm/sec
    :  FINE Tool_Offset
  31:  RO[7:Open Gripper]=ON
  32:L @PR[2:Pos1 Offset Ex] 100mm/sec
    :  FINE
  33:  CALL AA_HOME
[End]
```

```
                                                   22/34
  16:L @PR[5:Pos2 Offset Ex] 100mm/sec
    :  FINE Tool_Offset
  17:  !Release the pin
  18:  RO[7:Open Gripper]=ON
  19:  !Move to Pos2 without offset
  20:L @PR[5:Pos2 Offset Ex] 100mm/sec
    :  FINE
  21:  !Call another Program
  22:  CALL AA_HOME
  23:L @PR[5:Pos2 Offset Ex] 100mm/sec
    :  FINE
```

FANUC

Exercise 6

# Advanced Programing

FANUC

Exercise 6

# Advanced Programing

## Table of contents

## Background

Until now, you only have seen the really basics of programming. In exercise 4 and 5 you have already seen some more advanced techniques but we want to go a bit more into detail what you did there and even go deeper into what can be done with the robot.
This is important as the most applications afterwards will not be as simple as a simple move between two or more points.
As an example, the robot may distinguish between different situations in a program and then decide which sub-program to call.

## Equipment

For this exercise, you need:

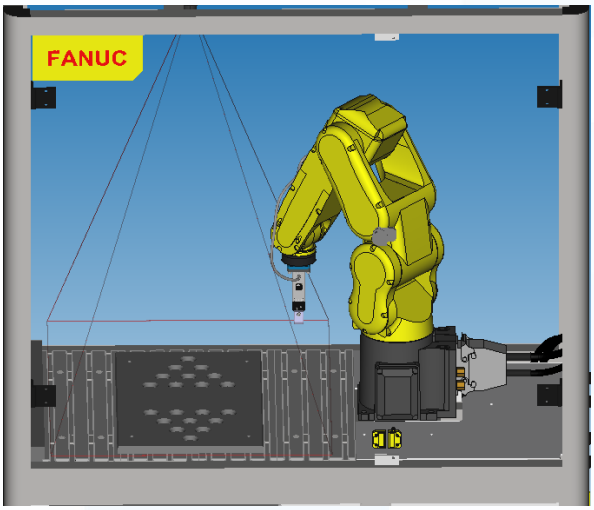
Education Cell

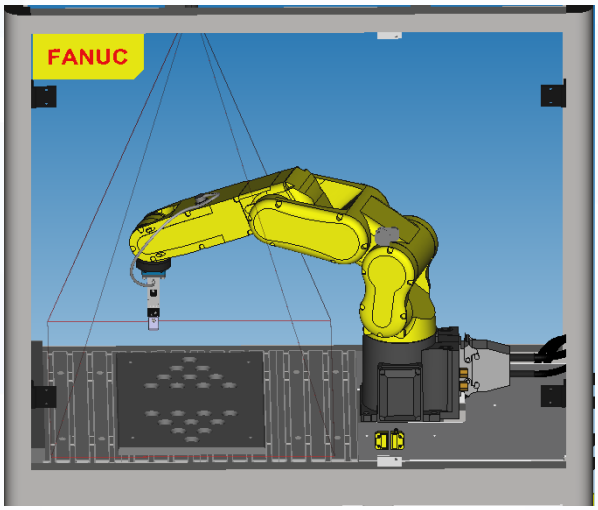
## First Program

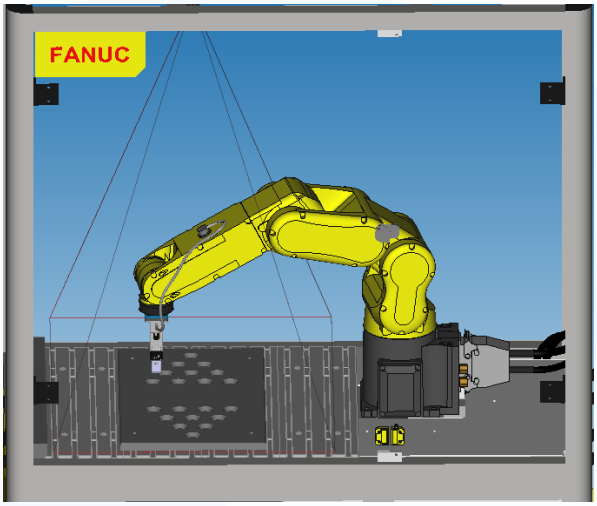
First thing we need to do, is to set the User Frame.
Press F1 Inst, then on page 2 number 4 Offset/Frames then UFRAME_NUM= and set it to 1 (World User Frame).

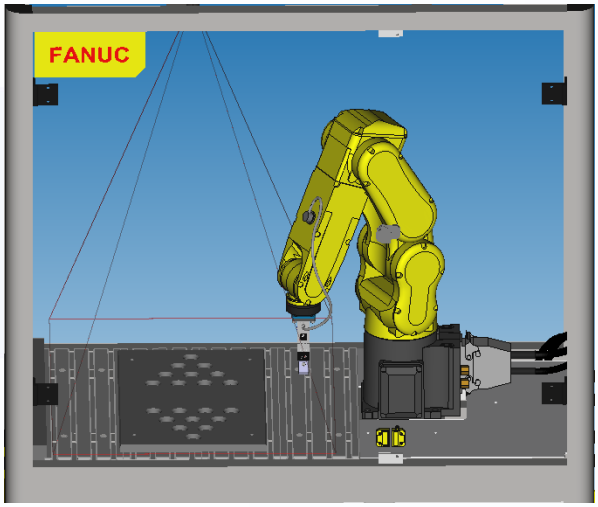
## First Program

Start by programming 4 different points in linear mode and 100mm/s speed. It is best to place them more or less in form of a rectangle. You should also use the same height. Start the program.



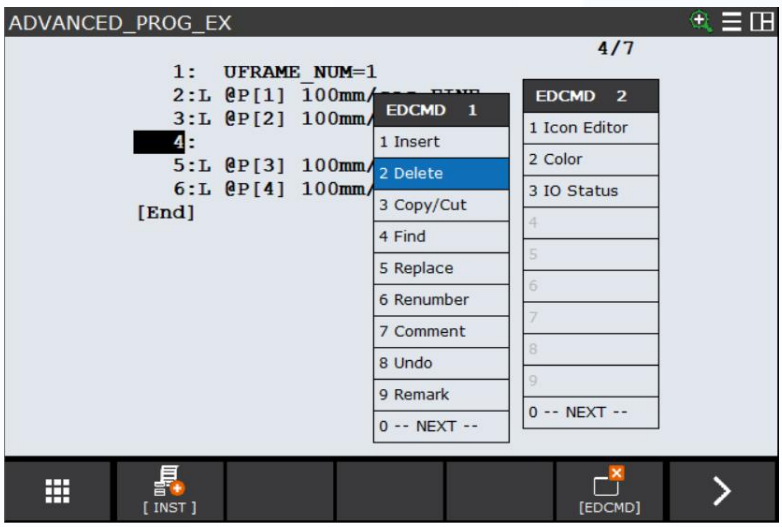



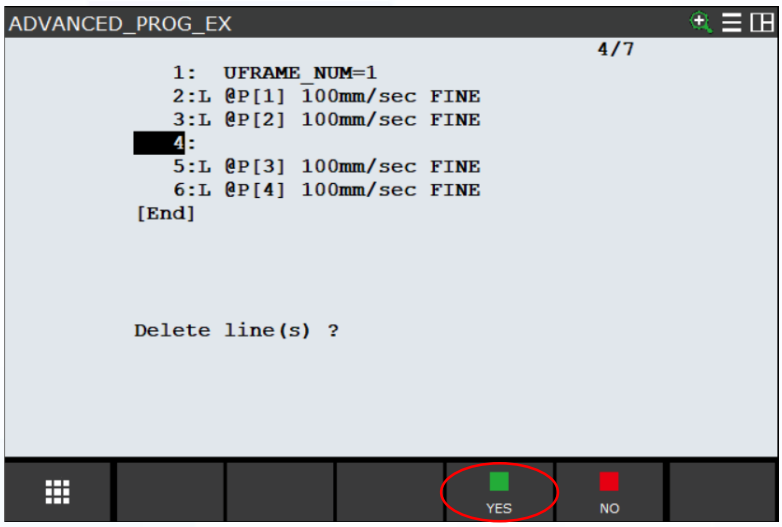
## Deleting

Now insert a row in between position 2 and 3. If no command is being set here, this line will not affect your program. To delete this line whether there is a command or not, push the EDCMD button (F5) and then select delete. You will be asked if you want to delete. Press F4.



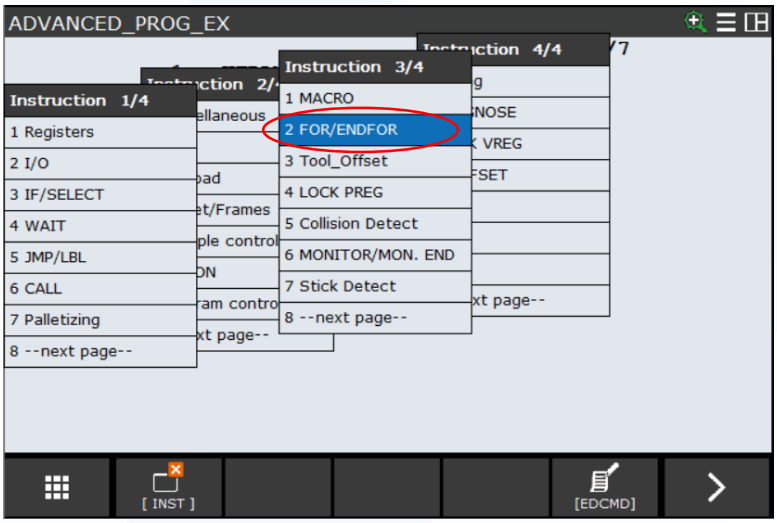

### Warning!

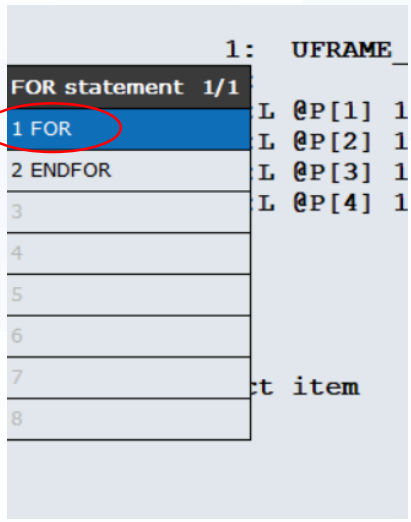
Deleting a line can cause serious damage to a program so before you delete al line make sure that you are completely aware of what this will change in your program.



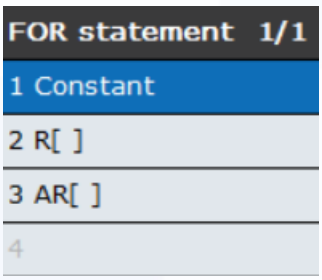
## For To Function

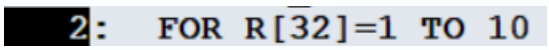
To let the program do the same motion or operation several times, there is the possibility to use “For to” functions.
So now we want to add a “For to” function.
Insert a new line above the motion instructions and press the “Inst” button and then input a For to.



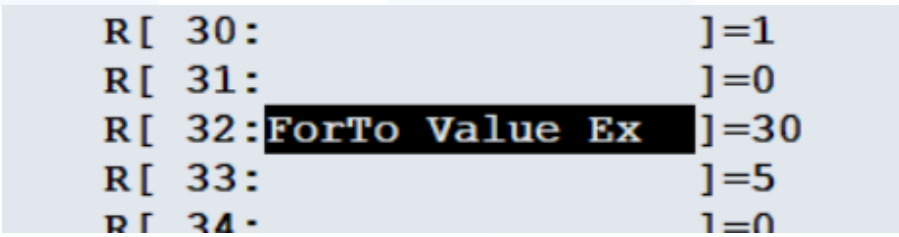

Choose a register. Make sure you use a register that has not been used in another program. (You can use R[32], this register should not be used by default)
Use the constant type twice and input 1 and 10.

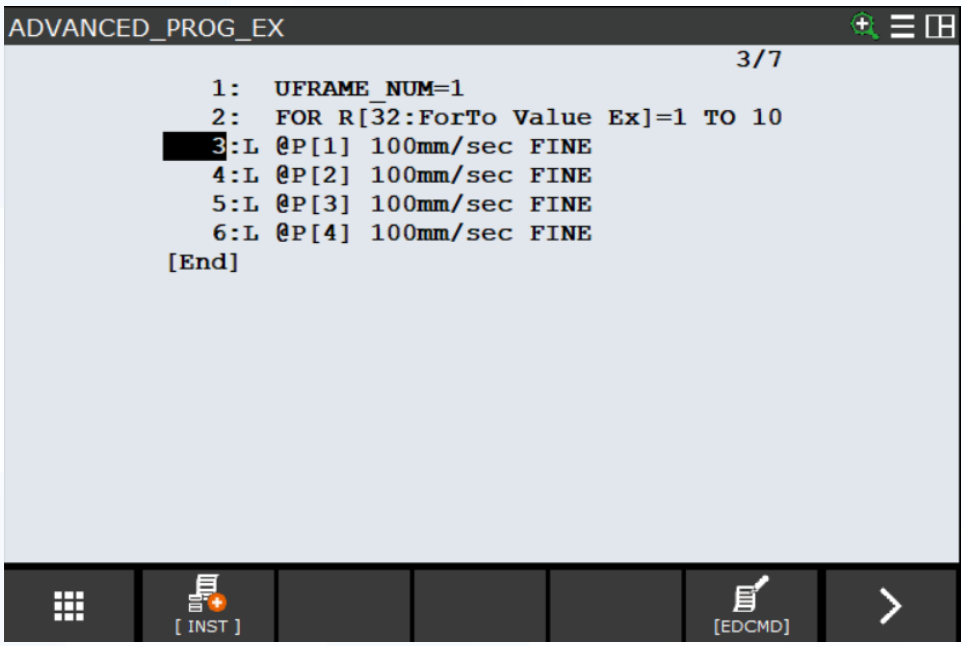


## For To Function

This means the program will repeat the procedure below and on every repetition, the register value will increase by one. So it will go from 1 to 10.
Now you should go to the register overview (press data button then as “type” choose Register) and rename R[32] in a way that you remember what it is used for. This name will also be indicated in the program.



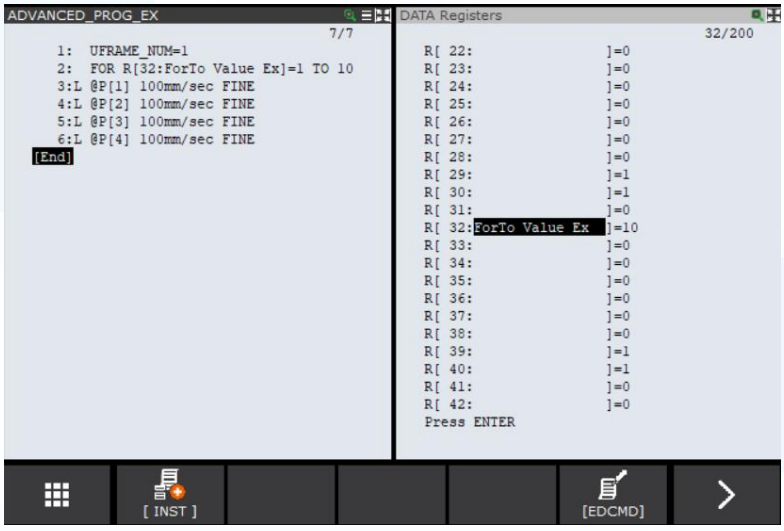



## For To Function

Start your program. You will see that the program will not start. This is due to the fact that the robot does not know how far the ForTo function is going. To resolve this, add and Endfor at the end of the program. (same procedure as for ForTo, only that you choose Endfor)

Now start the program again. This time the program should execute the movement 10 times.

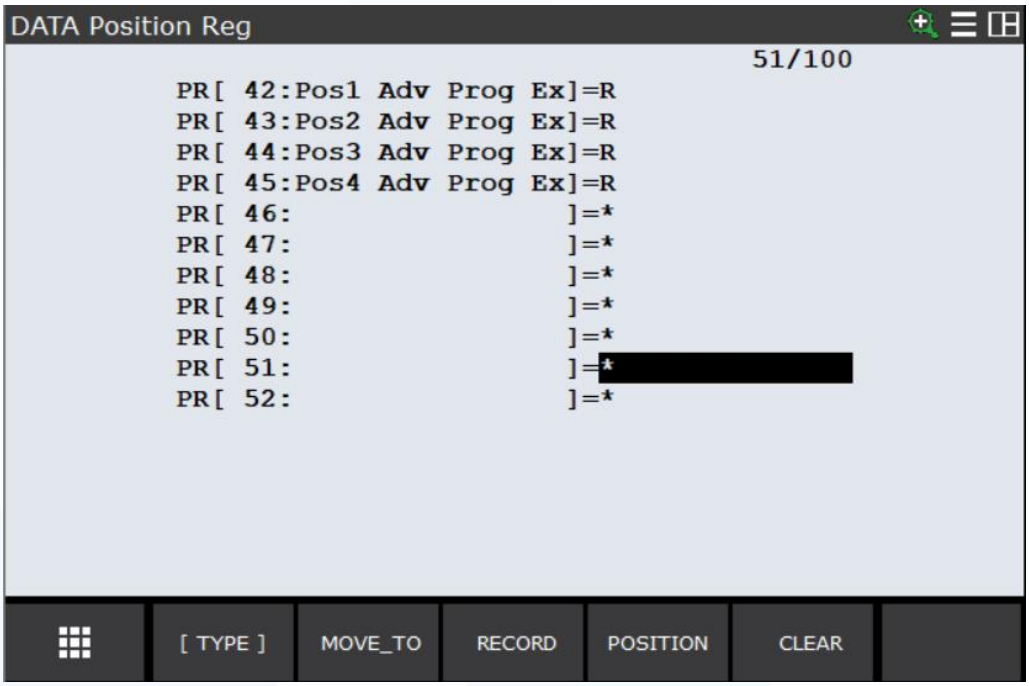
To see what happens in the register, we will add a second display. Press shift+disp and enter double. You now have two displays. To change between your different screens push the disp button without shift. Change to the second one and press Data button and change the type to Register. Now you can see the operations that your program does to the register. Start the program again and watch the register increase.




For further information about ForTo, refer to OM section 4.18 “For/Endfor Instruction”.

## Position Register

Now put the different positions into the PR to use them in secondary programs.



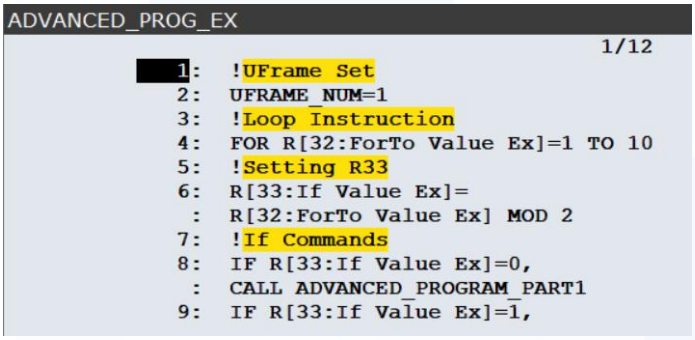

Create two new programs that you call Advanced_Program_part1 and Advanced_Program_part2.

## If Function

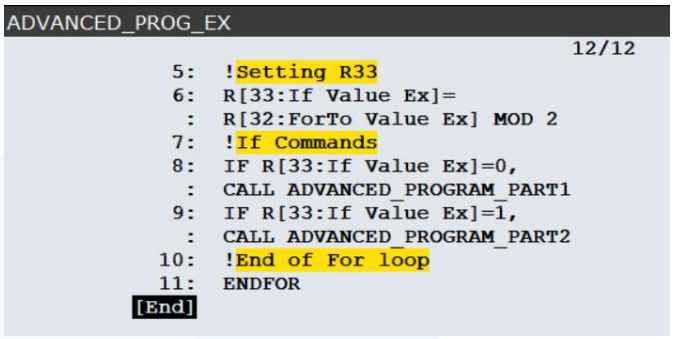
We want to let our program then do once the movement in the one direction and once in the opposite direction. Therefore we use an “If” function. It is used to check whether a declaration is right or not. If it is, it executes a given operation, if not, it just goes to the next command line.

In order to achieve our goal, insert a register instruction (F1 Inst then 1 Register) to calculate R[32] mod 2 (every second time, we will then get an input of 0). Input this value in R[33] (or another free register).

Now we need two if commands. One is calling Advanced_Program_part1, if the register equals 0, the other is calling Advanced_Program_part2 if the register equals 1.



```
ADVANCED_PROG_EX                                   1/12
   1:  !UFrame Set
   2:  UFRAME_NUM=1
   3:  !Loop Instruction
   4:  FOR R[32:ForTo Value Ex]=1 TO 10
   5:  !Setting R33
   6:  R[33:If Value Ex]=
    :  R[32:ForTo Value Ex] MOD 2
   7:  !If Commands
   8:  IF R[33:If Value Ex]=0,
    :  CALL ADVANCED_PROGRAM_PART1
   9:  IF R[33:If Value Ex]=1,
```



```
ADVANCED_PROG_EX                                  12/12
   5:  !Setting R33
   6:  R[33:If Value Ex]=
    :  R[32:ForTo Value Ex] MOD 2
   7:  !If Commands
   8:  IF R[33:If Value Ex]=0,
    :  CALL ADVANCED_PROGRAM_PART1
   9:  IF R[33:If Value Ex]=1,
    :  CALL ADVANCED_PROGRAM_PART2
  10:  !End of For loop
  11:  ENDFOR
[End]
```

For further information about if function, refer to OM section 4.7.4 “Conditional Branch Instructions” and 4.7.5 “If then/Else/Endif Statement”.

**PLEASE NOTE THAT**

Else functions are not included in the Education Cell package.

## Secondary Programs

The first program should execute the movement “the wrong way around” whilst the second program should do it “the right way around”.

```
ADVANCED_PROGRAM_PART1                                  5/5
          1:L @PR[44:Pos3 Adv Prog Ex]
           :  100mm/sec FINE
          2:L @PR[43:Pos2 Adv Prog Ex]
           :  100mm/sec FINE
          3:L @PR[42:Pos1 Adv Prog Ex]
           :  100mm/sec FINE
          4:L @PR[45:Pos4 Adv Prog Ex]
           :  100mm/sec FINE
        [End]
```

```
ADVANCED_PROGRAM_PART2                                  5/5
          1:L @PR[42:Pos1 Adv Prog Ex]
           :  100mm/sec FINE
          2:L @PR[43:Pos2 Adv Prog Ex]
           :  100mm/sec FINE
          3:L @PR[44:Pos3 Adv Prog Ex]
           :  100mm/sec FINE
          4:L @PR[45:Pos4 Adv Prog Ex]
           :  100mm/sec FINE
        [End]
```

## Jump Label

There is the possibility to use jump commands. The Jump LBL is used to jump to a specific label (defined in your program).
To use the jmp lbl, Press F1 (Inst) then 5 (JMP/LBL).
We have changed below Advanced_Program_Part1 to use it with Jump commands. As you can see it is very messy and not useful in this case. It should only be used as showcase. You can try and change your program to see that this function works and that there is no change in motion between the two possibilities.

```
ADVANCED_PROGRAM_PART1                                  5/5
          1:L @PR[44:Pos3 Adv Prog Ex]
            :  100mm/sec FINE
          2:L @PR[43:Pos2 Adv Prog Ex]
            :  100mm/sec FINE
          3:L @PR[42:Pos1 Adv Prog Ex]
            :  100mm/sec FINE
          4:L @PR[45:Pos4 Adv Prog Ex]
            :  100mm/sec FINE
        [End]
```

```
ADVANCED_PROGRAM_PART1                                 1/11
          1:  JMP LBL[1]
          2:  LBL[2]
          3:L @PR[45:Pos4 Adv Prog Ex]
            :  100mm/sec FINE
          4:  JMP LBL[3]
          5:  LBL[1]
          6:L @PR[44:Pos3 Adv Prog Ex]
            :  100mm/sec FINE
          7:L @PR[43:Pos2 Adv Prog Ex]
            :  100mm/sec FINE
          8:L @PR[42:Pos1 Adv Prog Ex]
            :  100mm/sec FINE
          9:  JMP LBL[2]
         10:  LBL[3]
        [End]
```

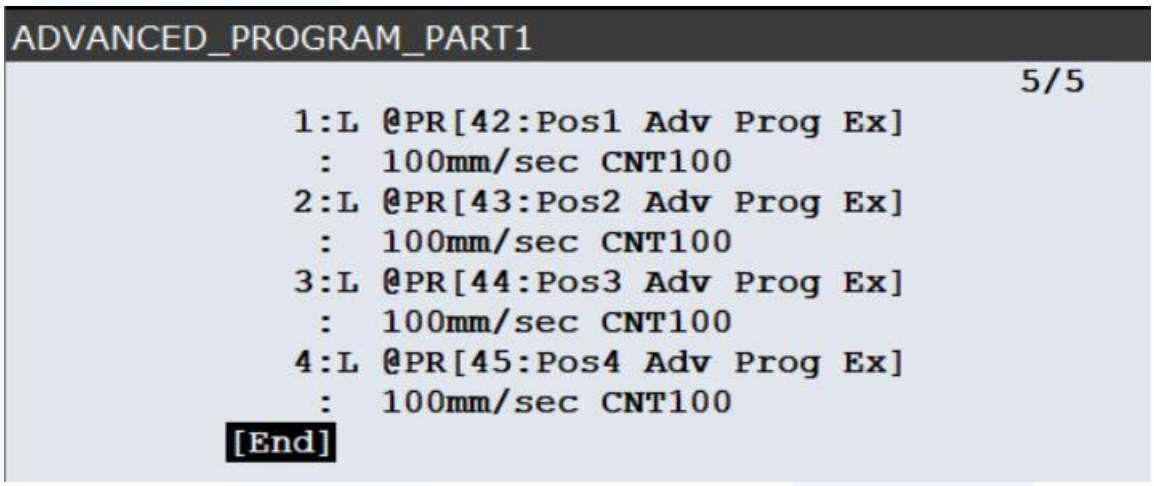
For further information about jump function refer to OM section 4.7.1 “Label Instruction” and 4.7.3 “Unconditional Branch Instructions”.

## CNT

Now we want to make our program more efficient meaning that the program takes less time. In order to do that we can use the CNT function instead of the Fine function. This does that the robot does not move exactly to one point but only moves in the environment of a point and then continues without stopping.

For more information about CNT motion refer to OM section 4.3.4 “Positioning Path”.

To see the difference, change the Advanced_Program_Part1 back to the right way around but change the motion setting from fine to CNT.



## CNT

When this program works without any problems, change the speed to 1000mm/s.

**Attention**: You MUST put override at 10 % or less and leave it there. Target of this setting is not to make the robot fast!

In your main program put in an extra line in front of the program and put in an override set function (inst-miscellaneous-override) and set it on 10(%). Don't change the override while the program is executing.

```
1:  !UFrame Set
2:  UFRAME_NUM=1
3:  !Set Override at 10%
4:  OVERRIDE=10%
5:  !Loop Instruction
```

Now run the program again and you will notice, that the path is much more round than before and also it is faster than with the fine setting.

This is due to the fact that the robot executes exactly the path, that it would do at 100% override (1000mm/s). The faster the speed of the motion the more CNT affects the motion.

## Wait Function

An additional option is to add a wait function.
This function will wait until a given time (by register or constant) and then continue in the next line.
Now add a wait function in every sub-program you called.

```
ADVANCED_PROGRAM_PART1                                  6/6
          1:L @PR[42:Pos1 Adv Prog Ex]
            :   1000mm/sec CNT100
          2:L @PR[43:Pos2 Adv Prog Ex]
            :   1000mm/sec CNT100
          3:L @PR[44:Pos3 Adv Prog Ex]
            :   1000mm/sec CNT100
          4:L @PR[45:Pos4 Adv Prog Ex]
            :   1000mm/sec CNT100
          5:   WAIT    2.00(sec)
        [End]
```

```
ADVANCED_PROGRAM_PART2                                  6/6
          1:L @PR[42:Pos1 Adv Prog Ex]
            :   1000mm/sec FINE
          2:L @PR[43:Pos2 Adv Prog Ex]
            :   1000mm/sec FINE
          3:L @PR[44:Pos3 Adv Prog Ex]
            :   1000mm/sec FINE
          4:L @PR[45:Pos4 Adv Prog Ex]
            :   1000mm/sec FINE
          5:   WAIT    2.00(sec)
        [End]
```

For more information about wait function refer to OM section 4.8.1 “Time-specified Wait Instruction” and 4.8.2 “Conditional Wait Instructions”.

## Recapitulation

In this exercise you should have learned the more advanced methods of programming making more complex programs which will help to develop more sophisticated programs and executions.
You should be able to use For to and If then clauses and call different sub-programs in your main program.
You should know the difference in path that makes your Tool if you use CNT function instead of FINE.
You should now the influence of speed on the CNT function.
You should now how to use the wait function.

FANUC

Exercise 7

# Input/Output

FANUC

Exercise 7

# Input/Output

## Table of contents

## Background

As already mentioned, I/O is used to let the robot communicate with other devices. The robot can get information from sensors or cameras or other devices through Input and give orders to motors, other robots etc. through Output.
For example, one robot gives an output to a second robot as soon as it has finished his work on a specific work part, so that the second robot can start it's own program on that part, and knows, that the first robot has finished with that part and is not in the way for further operations.
Another example is that a proximity switch can detect that a piece is at the end of a conveyor and then tell the robot to pick it up.

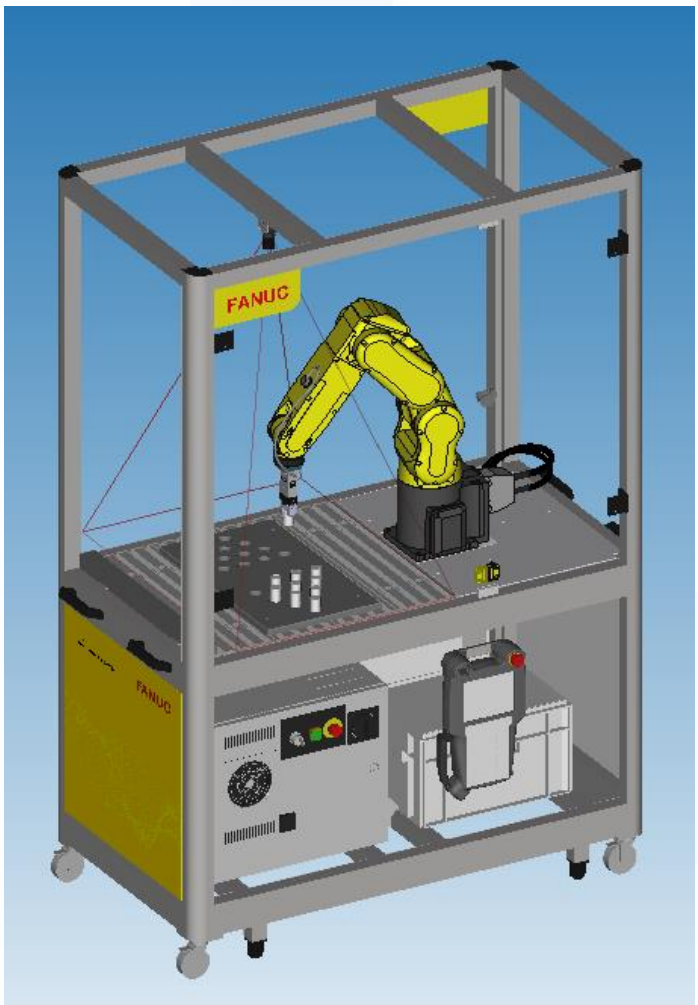
For more information about what is I/O refer to OM section 3.1 "I/O".

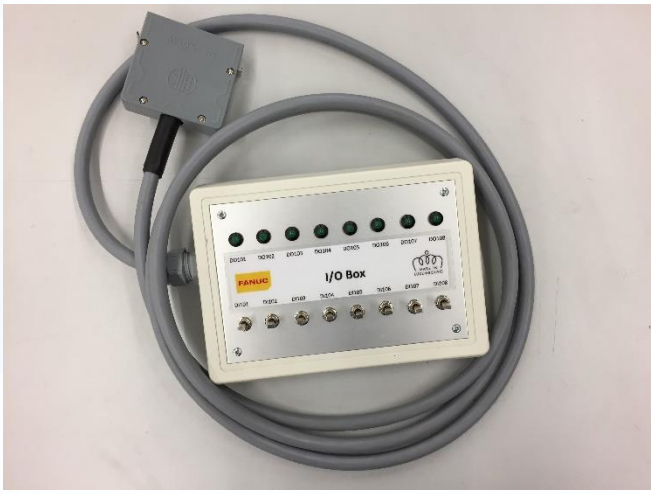
## Equipment

For this exercise, you need:

Education Cell

Input/Output Boxes (description is in the Appendix)




## Digital Inputs/Outputs

Firstly, we will go to the input and output window. Press the menu button, and then select 5 I/O.
You are now in the I/O menu. Change the type to Digital. To change between input and output, you can press F3 (IN/OUT). Go to output. As we only have outputs from 101 on, press the Item button and insert 101 then enter, so you will be redirected to the DO[101], our first LED on the Box. Try and change this output. You will see that it is not possible, as it is related with the robot being in the home position or not.


You can however change the other outputs and watch the LEDs going on and out again.
Now we will go to the inputs. Go directly to DI[101]. They should all be OFF. Now toggle one switch and see if the input switches to ON. See if all switches work without exception.
There is the possibility to “simulate” the input. Move the cursor to Sim and then press F4 Simulate. Now you can toggle the Input to On. If you Unsim it again, it will again be switched OFF (if there is no Input on that line).

For further information about simulate I/O refer to OM section 6.4.2 “Simulated I/O” and procedure 6-11 “Simulated input/output”.

## Digital Inputs/Outputs

Now we want to create a new program to use the new I/O.
Firstly, set UFrame to 1 and set Override to 10%.
Now you can call different programs for different inputs from your Digital I/O Box. We put an example below. Pay attention on what program you call, because it may be that you change the Override or the User frame during one program and you don't change it back for another program which can cause serious damage.

```
I_O_EX                                          1/16
    1:  !Set UFrame
    2:  UFRAME_NUM=1
    3:  LBL[1]
    4:  OVERRIDE=10%
    5:  IF DI[101:Switch 1]=ON,
     :  CALL ADVANCED_PROGRAM_PART1
    6:  IF DI[102:Switch 2]=ON,
     :  CALL ADVANCED_PROGRAM_PART1
    7:  IF DI[103:Switch 3]=ON,
     :  CALL ADVANCED_PROGRAM_PART2
    8:  IF DI[104:Switch 4]=ON,
     :  CALL ADVANCED_PROGRAM_PART2
    9:  IF DI[105:Switch 5]=ON,
     :  CALL AA_HOME
   10:  IF DI[106:Switch 6]=ON,
     :  CALL AA_HOME
   11:  IF DI[107:Switch 7]=ON,
     :  JMP LBL[2]
   12:  IF DI[108:Switch 8]=ON,
     :  JMP LBL[2]
   13:  WAIT    .10(sec)
   14:  JMP LBL[1]
   15:  LBL[2]
[End]
```

## Digital Inputs/Outputs

As a second exercise, we will do a little more complicated program. We want to “build” a stop watch.
The LEDs should indicate the time in Binary. (Pay attention that DO101 is used for indicating whether the robot is in HOME position, so this LED can’t be used for our program)
We need 3 switches to start, stop, and reset the timer.
Tip:

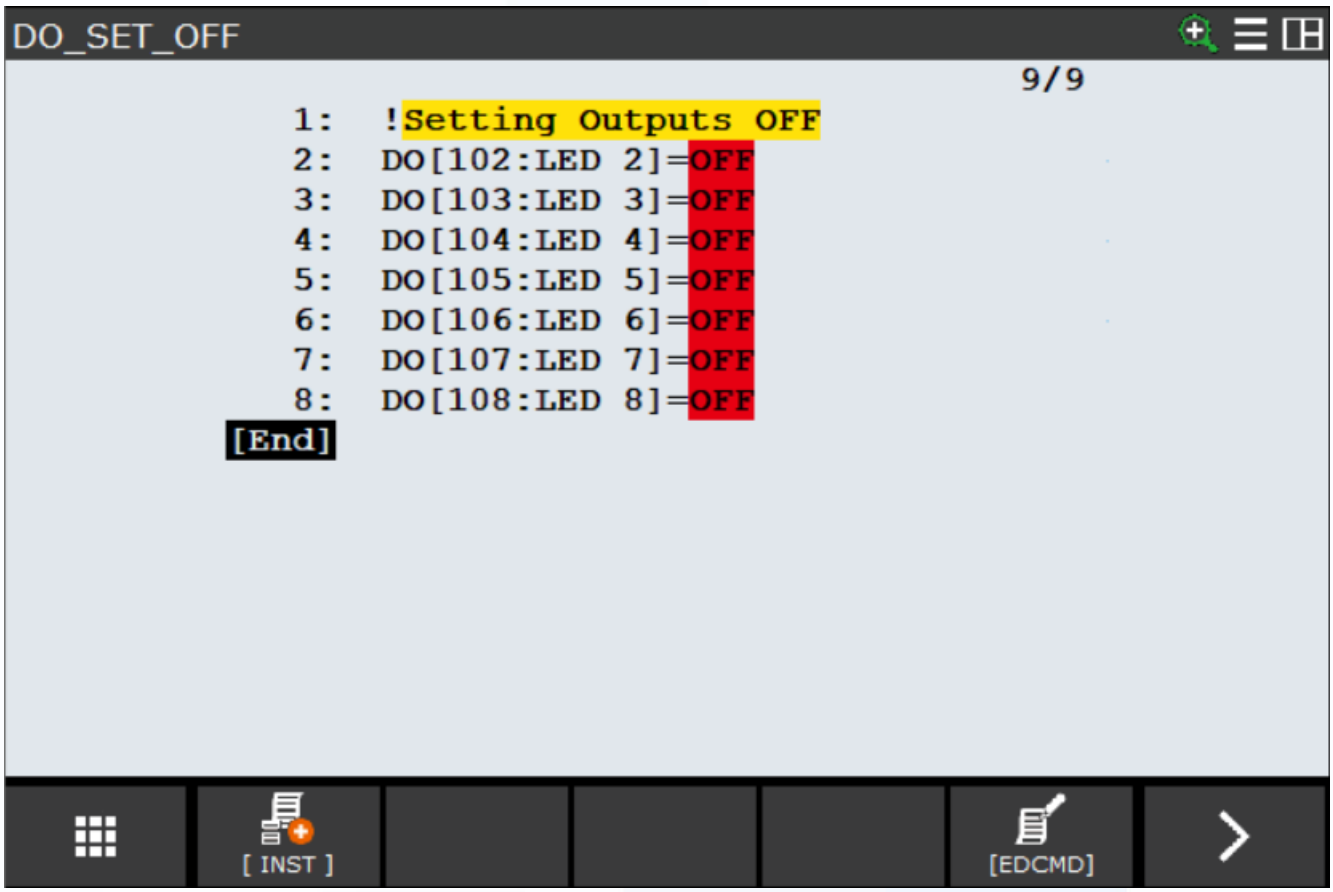
Use 3 different programs:

Set all DOs to OFF
Transforms a decimal number into binary and sets the outputs.
Main program which handles the different inputs and sets the timer (with Register).

Solution is on following pages.

## Digital Inputs/Outputs

Setting the Outputs on OFF



## Digital Inputs/Outputs

Transforming a decimal number into binary and setting the outputs.

DEC_TO_BIN 1/61

```
  1:  !Set Register
  2:  R[35:Bin Conv Reg]=R[34:Timer]
    :
  3:  !Reset Outputs
  4:  CALL DO_SET_OFF
  5:  !Start of Prog
  6:  LBL[9]
  7:  IF R[35:Bin Conv Reg]>=120,
    :  JMP LBL[1]
  8:  IF R[35:Bin Conv Reg]>=64,
    :  JMP LBL[2]
  9:  LBL[10]
 10:  IF R[35:Bin Conv Reg]>=32,
    :  JMP LBL[3]
 11:  LBL[11]
 12:  IF R[35:Bin Conv Reg]>=16,
    :  JMP LBL[4]
 13:  LBL[12]
 14:  IF R[35:Bin Conv Reg]>=8,
    :  JMP LBL[5]
 15:  LBL[13]
 16:  IF R[35:Bin Conv Reg]>=4,
    :  JMP LBL[6]
 17:  LBL[14]
 18:  IF R[35:Bin Conv Reg]>=2,
    :  JMP LBL[7]
 19:  LBL[15]
 20:  IF R[35:Bin Conv Reg]>=1,
    :  JMP LBL[8]
```

DEC_TO_BIN 22/61

```
 21:  JMP LBL[16]
 22:  !Counter max is 120 after substr
 23:  LBL[1]
 24:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-120
 25:  JMP LBL[9]
 26:  !Substr 64 turn on LED64
 27:  LBL[2]
 28:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-64
 29:  DO[108:LED 8]=ON
 30:  JMP LBL[10]
 31:  !Substr 32 turn on LED 32
 32:  LBL[3]
 33:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-32
 34:  DO[107:LED 7]=ON
 35:  JMP LBL[11]
 36:  !Substr 16 turn on LED 16
 37:  LBL[4]
 38:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-16
 39:  DO[106:LED 6]=ON
 40:  JMP LBL[12]
 41:  !Substr 8 turn on LED8
 42:  LBL[5]
 43:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-8
 44:  DO[105:LED 5]=ON
 45:  JMP LBL[13]
```

DEC_TO_BIN 47/61

```
 46:  !Substr 4 turn on LED4
 47:  LBL[6]
 48:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-4
 49:  DO[104:LED 4]=ON
 50:  JMP LBL[14]
 51:  !Substr 2 turn on LED2
 52:  LBL[7]
 53:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-2
 54:  DO[103:LED 3]=ON
```

DEC_TO_BIN 61/61

```
 53:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-2
 54:  DO[103:LED 3]=ON
 55:  JMP LBL[16]
 56:  !Substr 1 turn on LED1
 57:  LBL[8]
 58:  R[35:Bin Conv Reg]=
    :  R[35:Bin Conv Reg]-1
 59:  DO[102:LED 2]=ON
 60:  LBL[16]
[End]
```

## Digital Inputs/Outputs

I_O_TIMER 1/29

```
    1:  !Reset the Timer
    2:  R[34:Timer]=0
    3:  LBL[2]
    4:  !DI101 starts timer
    5:  IF DI[101:Switch 1]=ON,
     :  JMP LBL[1]
    6:  !DI102 stops program
    7:  IF DI[102:Switch 2]=ON,
     :  JMP LBL[5]
    8:  !DI103 resets timer
    9:  IF DI[103:Switch 3]=ON,
     :  JMP LBL[4]
   10:  !Wait for not ovrloadng Processor
   11:  WAIT     .10(sec)
   12:  !Jump to beginning
   13:  JMP LBL[2]
   14:  !Add 1sec to Reg and Output LED
   15:  LBL[1]
   16:  WAIT    1.00(sec)
```

I_O_TIMER 24/29

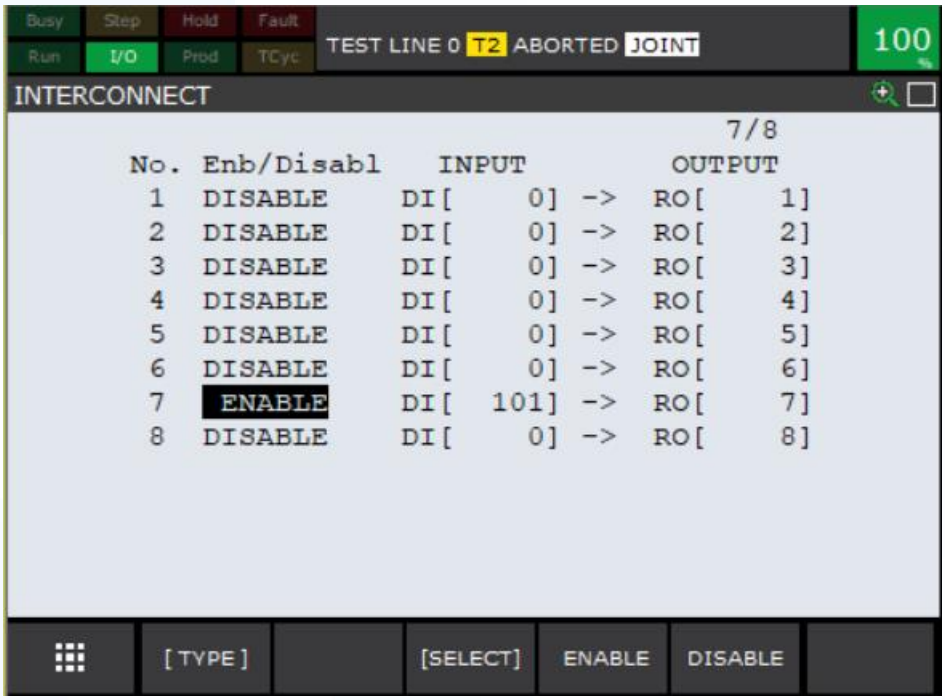
```
   17:  R[34:Timer]=R[34:Timer]+1
   18:  CALL DEC_TO_BIN
   19:  IF DI[102:Switch 2]=ON,
     :  JMP LBL[5]
   20:  IF DI[101:Switch 1]=OFF,
     :  JMP LBL[2]
   21:  JMP LBL[1]
   22:  !Reset Timer
   23:  LBL[4]
   24:  R[34:Timer]=0
   25:  CALL DO_SET_OFF
   26:  JMP LBL[2]
   27:  !End of Program
   28:  LBL[5]
[End]
```

## Interconnect

With interconnect you can give the robot an Input which will trigger an Output regardless of when. (be careful when you use interconnect).
In this example we will use the DI101 to open the gripper. If DI101 is on the gripper will open and if DI101 is off the gripper will close.

To set up the interconnect press “MENU”, ‘5 I/O’ and press ‘9 Interconnect’. We have to be in the screen with DI as input and RO as output. Press F3 “Select” and then ‘2 DI->RO’.

Go to number 7 and for DI enter the number 101. The interconnect is disabled so go to disabled and press F4 ‘Enable’ to enable (with F5 you can disable the interconnect again).




For further details go to the OM section 3.6 “I/O Connection Functions”

## Interconnect

Now when you switch the DI101 on the gripper will open and if the DI101 is off the gripper will close.

However this can bring unforeseen results with it.

To demonstrate it you should take all cylinders out of the Education Cell and start the program AAA_DEMO in ‘AUTO’ mode.
When the program is executed turn the DI switch on and off to see what happens, especially when the robot is just about to grab or release a cylinder.

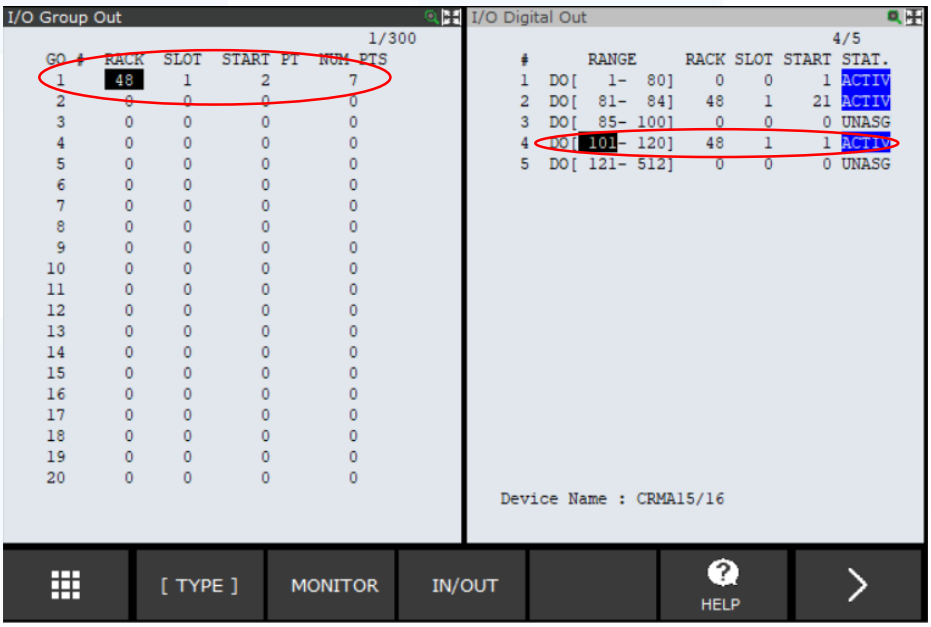
## Group I/O

The Group I/O is used when 2 or more Inputs or Outputs are used simultaneously. The value of a Group I/O is binary number (1, 2, 4, 8, 16, 32, …).
For this exercise we will use the DO[102] to DO[108].
First of all we want to have a split screen to have it easier to set up. Press “SHIFT” and the “DISPLAY” key and select ‘2 Double’.

On the second screen go to the DO screen. Then press F2 ‘Config’.

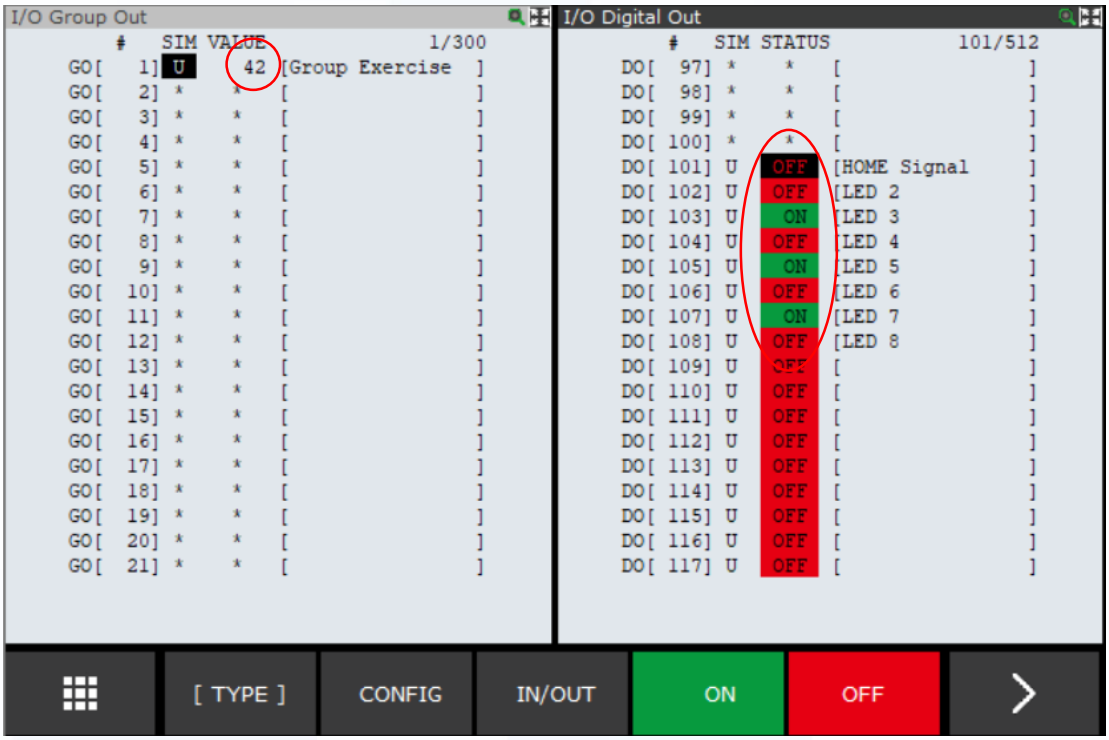
On the first screen, press “MENU”, ‘5 I/O’ and then ‘5 Group’. We need to be in Output (GO). Then press F2 ‘Config’.

At line 4 on the second screen you see DO 101 to 120. That line gives us the information we need to set up the Group.
We use GO[1], in ‘Rack’ put in the number ’48’, in ‘Slot’ number ‘1’. The start point is DO[102] so put in ‘START PT’ the number ‘2’ and in ‘NUM PTS’ number ‘7’ because we go to DO[108].



## Group I/O

Now we have to cycle the power.
Use the split screen again and see which value GO[1] has when you turn the different LEDs on.

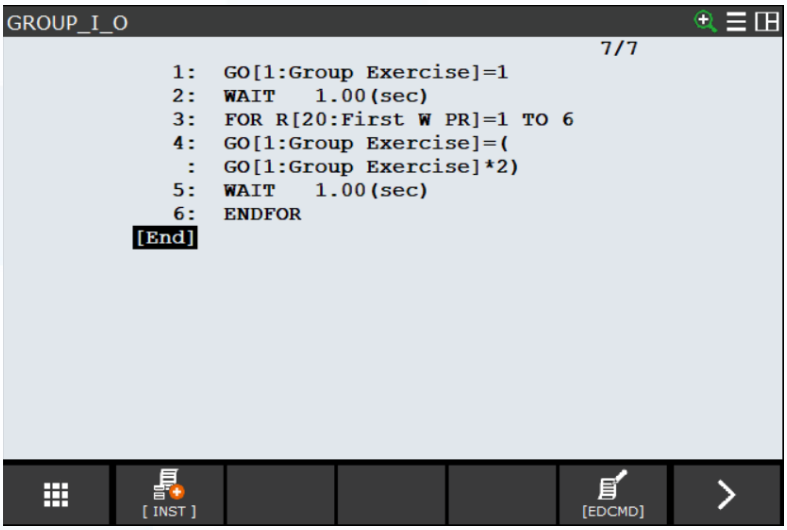



Now we want to program a LED show. In this program we use the GO[1] to turn the different LEDs on.

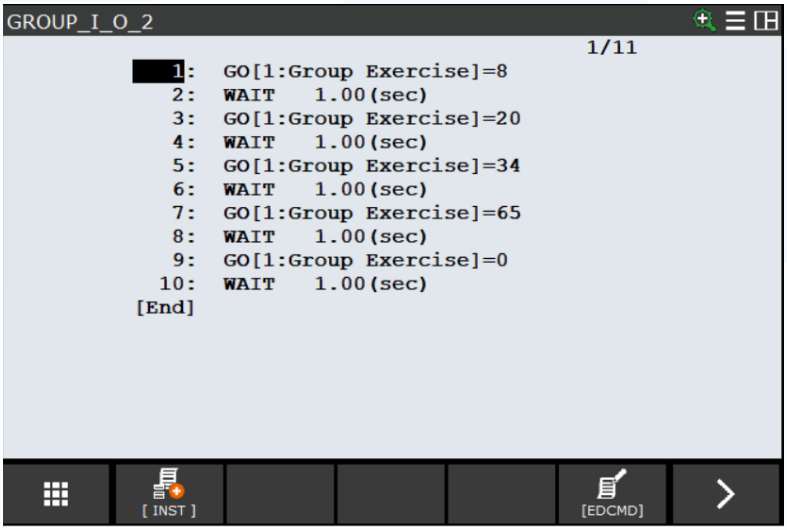
For more information on ‘Group I/O’ see OM section 3.1.2 “Group I/O” and procedure 3-2 “Configuring Group I/O”.

## Group I/O

The first program will turn the LEDs on like a wave from DO[102] to DO[108].




In the next program the DO[105] will be the epicentre and then the other LEDs will turn on.



## Group I/O

As you can see this makes it a lot faster to program and the LEDs will turn on/off simultaneously. If you always program with DO[…]=ON then it can be that there will be a small lag and will not look as good.

You can use also the Group Input for the switches. GI works the same as GO only with inputs instead of outputs.

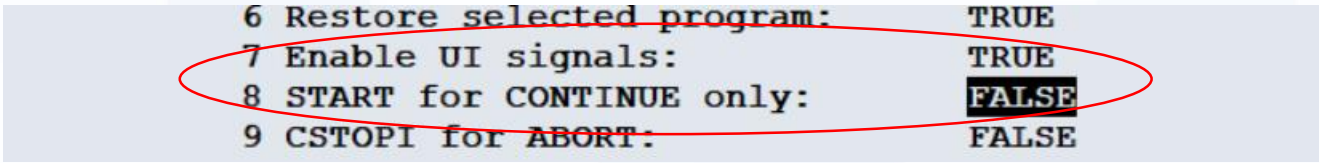
## User Inputs/Outputs

We now start to operate with our UOP Box (User Operator Panel). This should emulate a panel like the ones used in industry, to input the very basic commands such as start and program selection. Program selection is made via the PNS switches. A program that you want to be called, has to be named PNS followed by a 4 digit number (e.g. PNS0011). PNS is read as a binary number when no program is on hold and the start button is pressed.

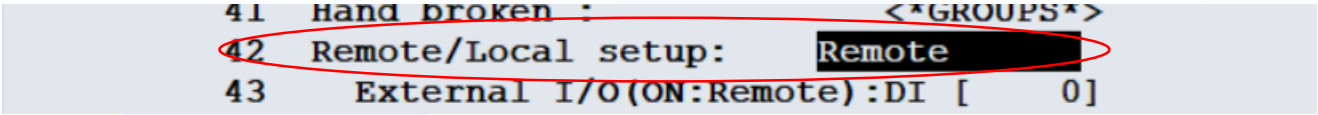
For more information about UOP I/O refer to OM section 3.3 “Peripheral I/O” and procedure 3-5 “Assigning Peripheral I/O”.
Note that in our case we only have the simple assignment for UOP (Simple CRMA16).

## User Inputs/Outputs

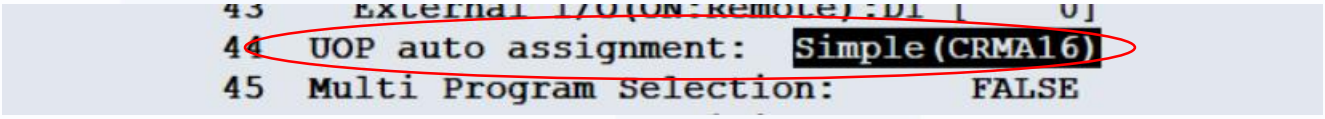
Firstly we need to enable UOP to be able to command the robot from the UOP. Press the menu button, then on the second page System (6) then Config(6).
7 Enable UI Signals has to be set on True and 8 Start for continue only on False.




42 Remote/Local Setup set on Remote




44 UOP auto assignment set on Simple(CRMA16)




Now cycle power to make the settings valid.

For further information about setting up UOP refer to system config menu in OM section 3.15 “System Config Menu” and procedure 3-33 “Setting the System”.

## User Inputs/Outputs

By now we start with a simple example of starting a program from the UOP.
Select a program with the teach pendant. Put the Controller on Auto mode and turn the teach pendant off. Now Enable the UOP (ENBL) and press the Hold button on UOP (Hold signal is active low, so should normally be high). Press the Start button. Notice that the program will only start at the release of the Start button.
You have now started your first program from a UOP.
We now want to go a step further and be able to select a program from the UOP. In order to do that we need programs to be called PNS plus a for digit number. Go ahead and copy a program and name the copy PNS0001. Now before you toggle the start switch, toggle the PNS selector switch (in our case PNS1). The program PNS0001 should be started.

For further information about the PNS selecting function refer to OM section 3.8.2 “Program Number Selection (PNS)”, procedure 3-9 “Setting the PNS function” and procedure 6-15 “Automatic operation by program number selection”.

## User Inputs/Outputs

While the robot is operating you will notice the LEDs that are on. First LED, ENBL goes on if operator panel is on. The last LED, BUSY is on while a program is running. Then there are the two LEDs in the middle. The second in the row is the FAULT LED and third is the BATALM. BATALM should not occur while operating the robot, so we will not go to much into detail, you can for more information refer to the manual.
The FAULT LED however can occur and we want to simulate one. In order to do this, start a program. While the program is running open the door of the cell. The robot stops and the FAULT LED is turned on. This is due to the circuit breakage of the fence. Close the door again, press the reset button (on UOP) and resume the program (with start on UOP).

## Multiple Programs in Parallel

Lastly we want to show you, that you can also run several programs in parallel instead of only one program after another. This is particularly practical if you want to move the robot as well as let it command exterior equipment as well.
To show this, we will do the moving program (10 times the square movement) at the same time than the timer program. Again, this is not very important nor practical in our case, but there are different situations where this can be very practical. Feel free, after this exercise of thinking about such a scenario and do the programming yourself. After you have done all the exercises including this one, you should be able to do this.
As we may want to use that program again later without those additions, make a copy of that program which you call Advanced_Prog_Ex_IO. Continue on working on that program.

For more information about Run command refer to OM section 4.16.1 “Program Execution Instruction”.

## Multiple Programs in Parallel

Go into the Advanced_Prog_Ex program. Add a new line behind the override instruction. Add a Run Command (Instr second page 5 Multiple Control) and Run I_O_Timer.
Try and start your program.
The program will not start and the TP will output “PROG-040 Already locked by other task”.
To resolve this you need to change the Group Mask of the programs used in the Run command (I_O_Timer, DO_Set_OFF, Register_To_Binary) from 1 to *. This can be done by pressing F2 Detail.

```
Program name:
 1 DEC_TO_BIN
 2   Sub type:        [None              ]
 3   Comment:         [                  ]
 4   Group mask:      [*,*,*,*,*,*,*,*   ]
 5   Write protect:   [OFF               ]
 6   Ignore pause:    [OFF               ]
 7   Stack size:      [     500          ]
 8   Collection:      [                  ]
```

For more information about Group Mask, refer to OM section 4.1.4 “Group Mask”.

## Multiple Programs in Parallel

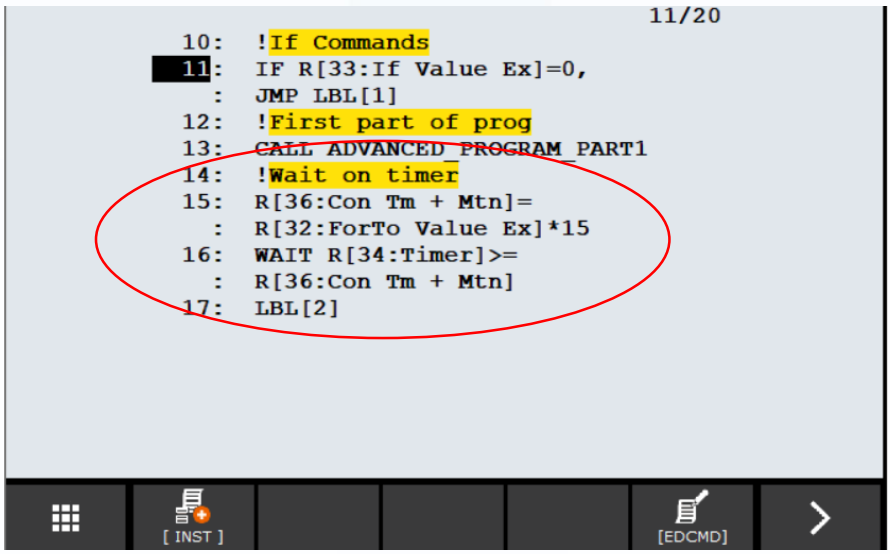
Now your program should start and be able to run both programs in parallel meaning that the robot is moving while you can “play” with the stopwatch.

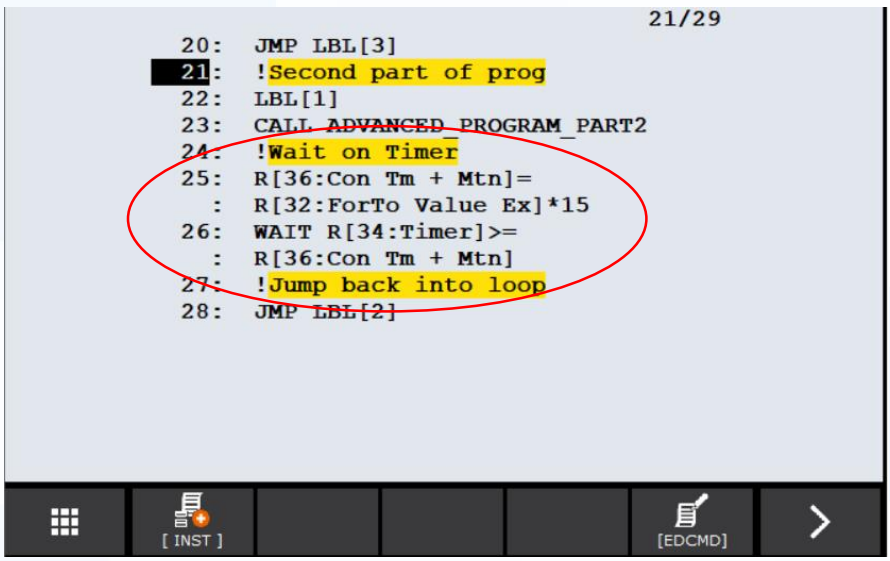
As mentioned, you are able to “play” with the stopwatch meaning that it has no use in our program. As the robot is not built to play with, we are now going to change the program so that the one program is interacting with the second one, so that the programs do run independently, but after all are connected to each other in some way.

We have already input a wait instruction after the program has achieved one tour. We now want to change that wait signal so that it does not wait exactly 2 seconds, but that it waits until a specific register value used in the second program is achieved. In doing so, the motion program always waits on the non motion program so that they are connected.

## Multiple Programs in Parallel

In the following, we have put the changed parts of the program.



## Multiple Programs in Parallel

Run the program and pay attention whether the program stops if the counter is not at a multiple of 15. You can try and stop the counter or reset it an start again. The program should wait for the right time and then continue with the next motion.
After the motion finishes, the program will continue to run, because the Timer program does not have a stop. To change that, use a register that will be checked by the timer program and end that program when the motion program is ended.

## Recapitulation

In this exercise, you should have learned the principals of Input and Output.
You should know the different options there are with inputs and outputs and be able to create a program that executes two different programs at a time (only one being a motion program(!)).
You should be able to set up a User Operator Panel and run the robot through this panel.
You should be able to connect peripheral devices to the robot and set the Inputs and Outputs to use those devices while robot operation.

## Appendix

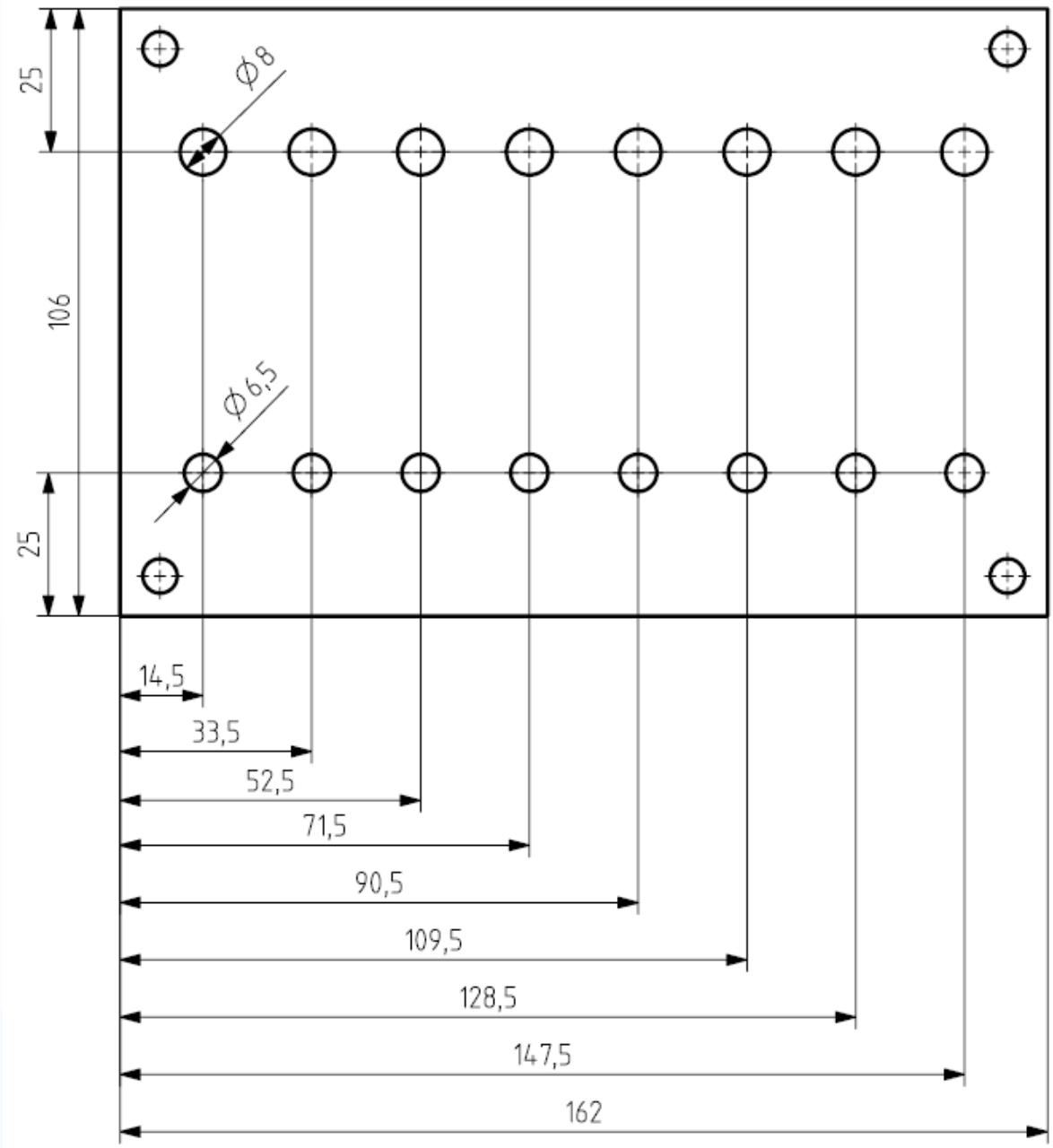
In order to be able to do this exercise you need two input/output boxes.
Those parts are all from the RS website as well as the references. (www.rs-components.com)
Part list needed for these boxes are below:

- 8x 8mm green LED 24V (ref. nbr RS 206867)
- 2x Aluminium sloped panel case 190x138x47 (ref.nbr. RS 505820)
- 2x 8mm red LED 24V (ref.nbr.RS 206845)
- 2x 8mm yellow LED 24V (ref.nbr.RS 206463)
- 16x On Off (On) switches 4A 30V (ref.nbr.RS 1035502)
- 2x Honda Connector MR-50M (Fanuc ref.nbr. 816052)
- 2x Honda Connector Case MR-50M (Fanuc ref.nbr. 836896)
- 6m Flex cable (min. 18 cables, 3m per box)

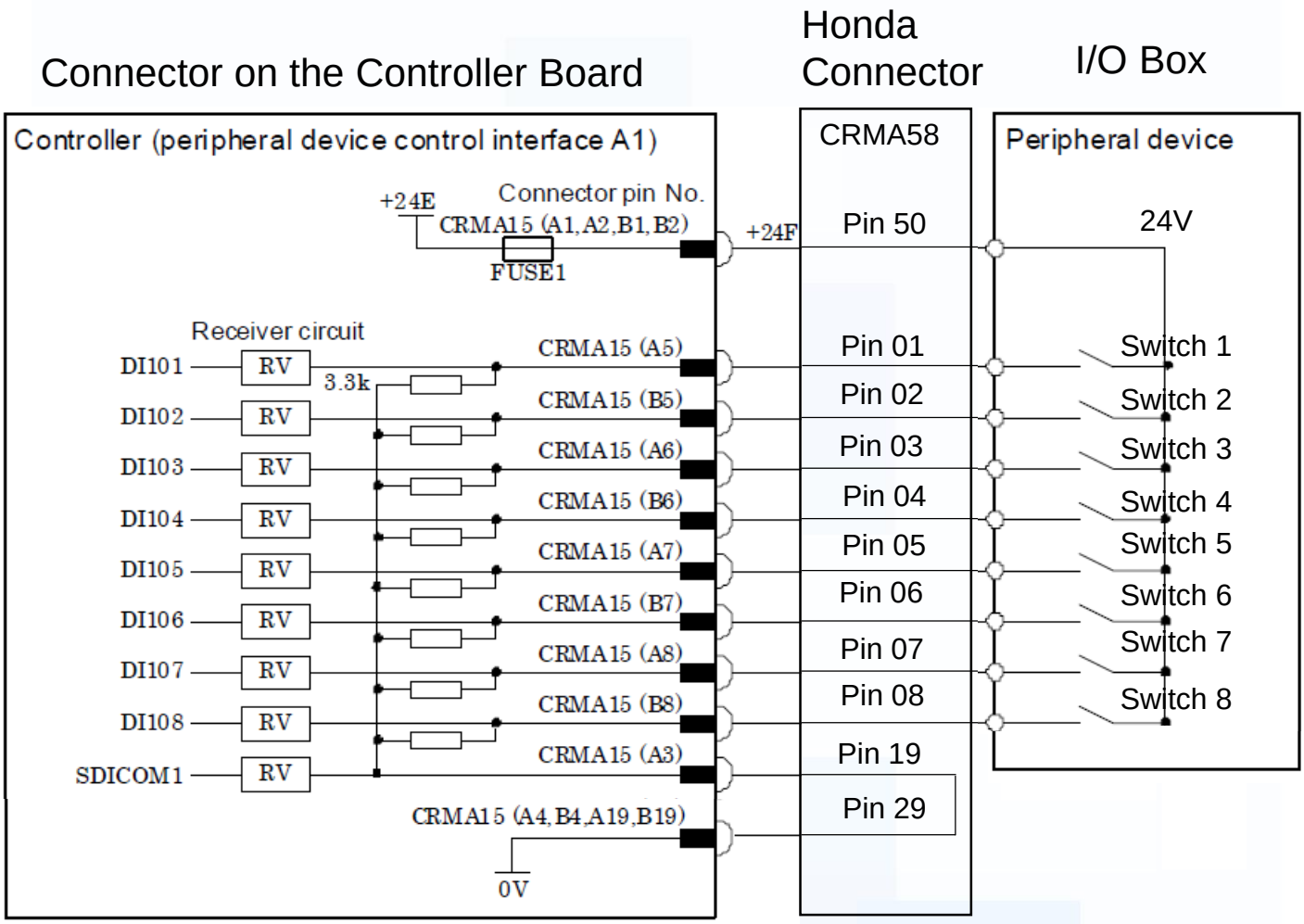
## Appendix

You start by drilling holes in your case for the LEDs and the switches.
If you use our recommended case, you can take the plans below for the drilling.
In the first box, put in 8 switches and 8 Green LEDs (you can take a colour of your likings)
In the second one, put in also 8 switches, but there you only need 4 LEDs and we would recommend to use 2 red and 2 yellow LEDs.
If you use the recommended box, you can use the following drawing for the dimensions and the positions for the holes that need to be drilled in the aluminium board.

## Appendix



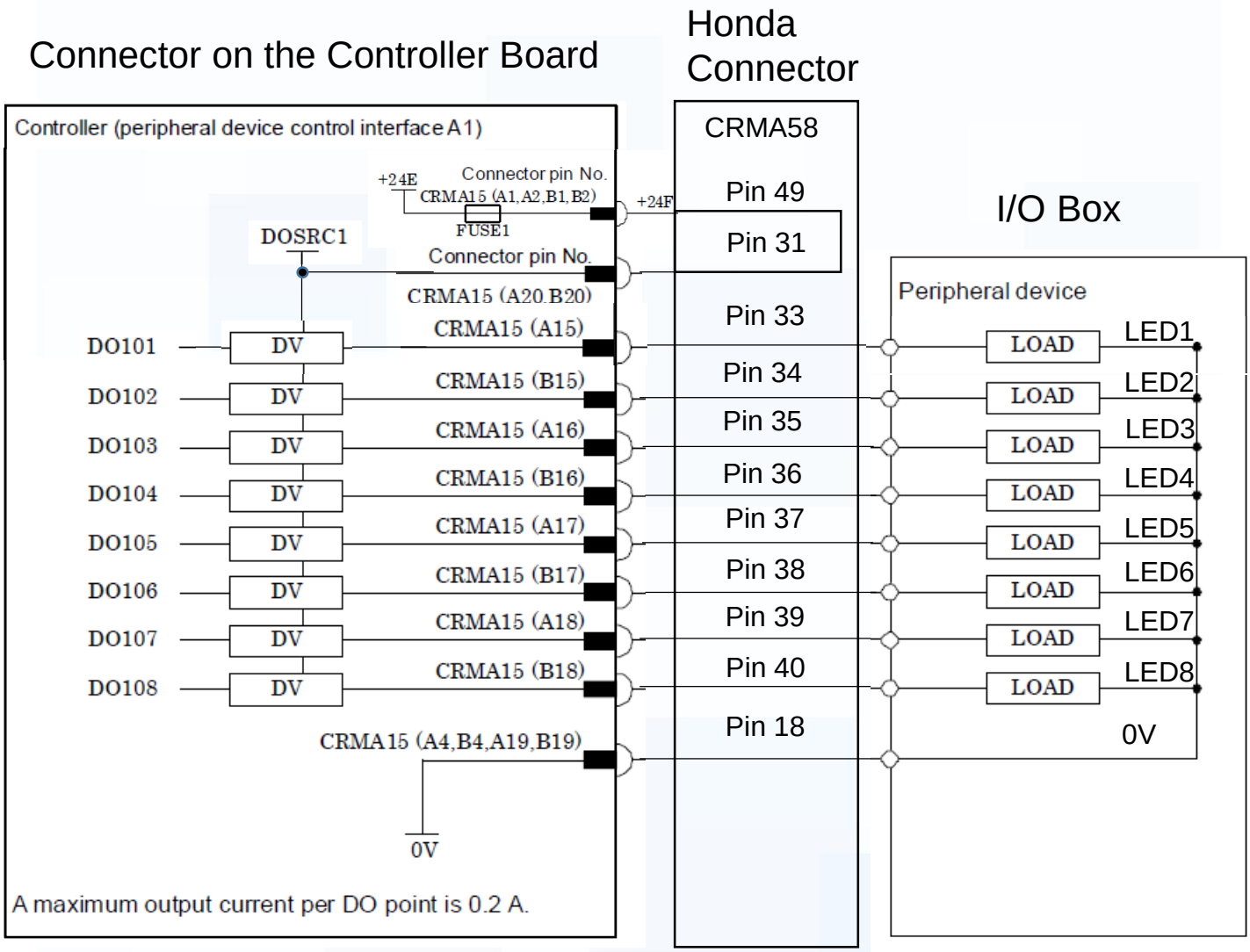
## Appendix




The 0V to the SDICOM1 connection is made directly in the Honda connector (CRMA58).
Connect the 24V to all the On and (ON) positions of the switches.
Then connect the OFF positions of all the switches to the cable. Make sure to note which cable belongs to what switch.

For complete plans refer to R-30iB Mate Controller MAINTENANCE MANUAL section 4.3 “INTERFACE FOR PERIPHERAL DEVICES”.

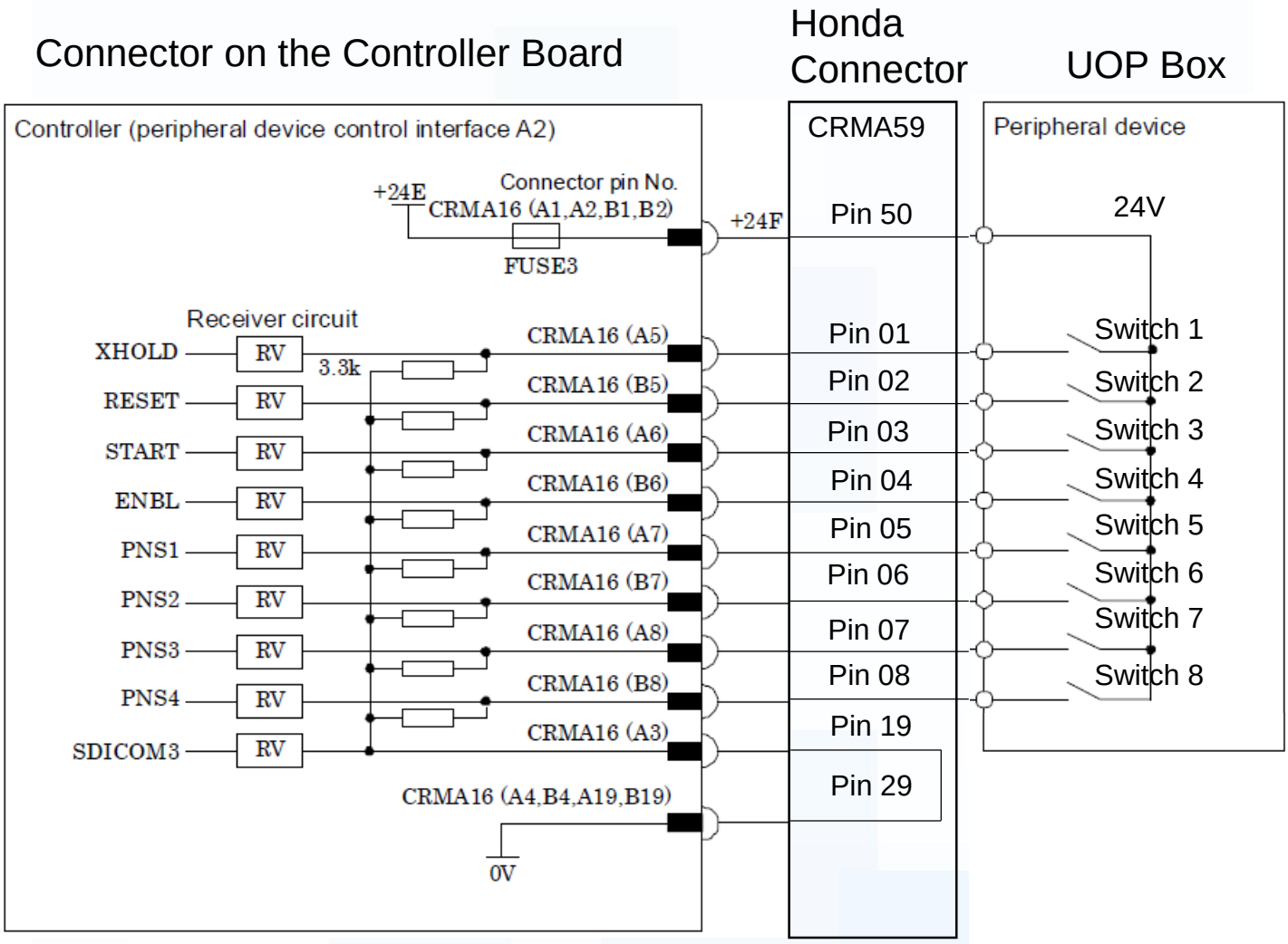
## Appendix




Then connect the LEDs in the first box. This time connect all the 0V together and connect the 24V separately with the LEDs. Connect the 24V to the DOSRC1 as shown above.

## Appendix

Connector on the Controller Board

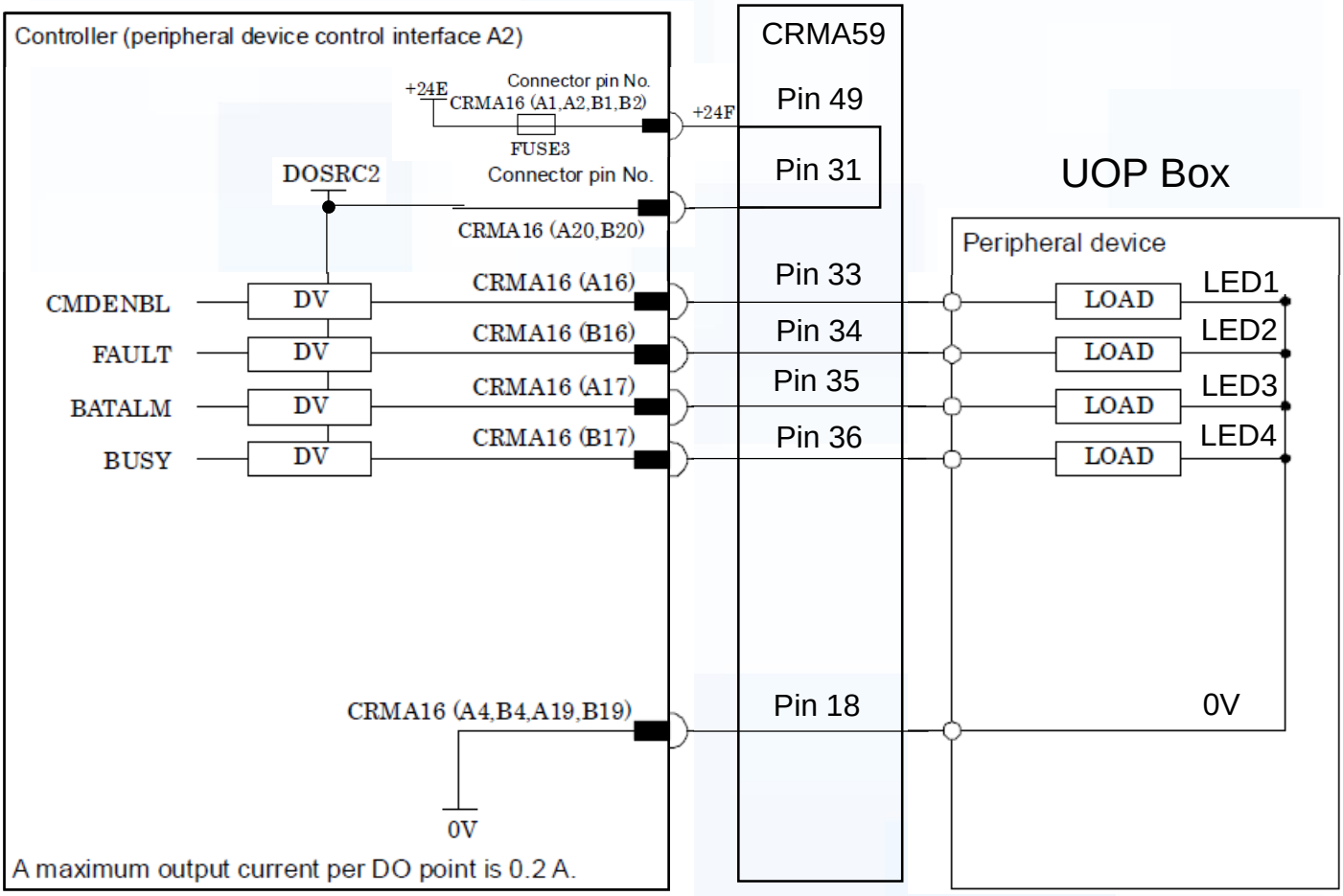



The second box is done exactly the same than the first box and we would also recommend you to use the same cable combination than in the first box to avoid confusion later on. This will also allow you to change the connectors without having short circuits.

## Appendix

Connector on the Controller Board

Honda Connector




Again the LEDs in Box 2 connect the same as in box 1 only that this time, there are only 4 LEDs as opposed to 8 for the first box. Do not forget to connect 24V to DOSRC2.

## User Inputs/Outputs

| R30iB Mate Controller CRMA15/CRMA16 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| CRMA 58 | | | | | | CRMA 59 | | | | |
| DI | Signal | Wire | Pin # | Input | | UOP In | Signal | Wire | Pin # | Input |
| | S1 | Blue | 1 | DI 101 | | | S1 | Blue | 1 | XHOLD |
| | S2 | Red | 2 | DI 102 | | | S2 | Red | 2 | RESET |
| | S3 | Purple | 3 | DI 103 | | | S3 | Purple | 3 | START |
| | S4 | Gray | 4 | DI 104 | | | S4 | Gray | 4 | ENBL |
| | S5 | Pink | 5 | DI 105 | | | S5 | Pink | 5 | PNS1 |
| | S6 | Green | 6 | DI 106 | | | S6 | Green | 6 | PNS2 |
| | S7 | White | 7 | DI 107 | | | S7 | White | 7 | PNS3 |
| | S8 | Yellow | 8 | DI 108 | | | S8 | Yellow | 8 | PNS4 |
| | / | Brown | 50 | 24V | | | / | Brown | 50 | 24V |
| | | | | | | | | | | |
| | | | | | | | | | | |
| DO | Signal | Wire | Pin # | Output | | UOP Out | Signal | Wire | Pin # | Output |
| | LED1 | Gray/Pink | 33 | DO 101 | | | LED1 | Gray/Pink | 33 | CMDENBL |
| | LED2 | Red/Blue | 34 | DO 102 | | | LED2 | Red/Blue | 34 | FAULT |
| | LED3 | Brown/Green | 35 | DO 103 | | | LED3 | Brown/Green | 35 | BATALM |
| | LED4 | White/Yellow | 36 | DO 104 | | | LED4 | White/Yellow | 36 | BUSY |
| | LED5 | Yellow/Brown | 37 | DO 105 | | | / | White/Gray | 18 | 0V |
| | LED6 | White/Green | 38 | DO 106 | | | | | | |
| | LED7 | White/Pink | 39 | DO 107 | | | | | | |
| | LED8 | Pink/Brown | 40 | DO 108 | | | | | | |
| | / | White/Gray | 18 | 0V | | | | | | |
| | | | | | | | | | | |
| Internal Connection | Signal | Pin # | Signal | Pin # | | Internal Connection | Signal | Pin # | Signal | Pin # |
| | SDICOM1 | 19 | 0V | 29 | | | SDICOM3 | 19 | 0V | 29 |
| | DOSRC1 | 31 | 24V | 49 | | | DOSRC2 | 31 | 24V | 49 |

## User Inputs/Outputs

To plug our cases on the robot controller, the first thing you need to do is to turn of the power of the controller and unplug the cell.
The cell has to stay unplugged until the door of the controller is closed again!
Open the door of the controller. Pull in the cables on the right side of the controller. The plugs for the Honda connectors are on the right. Make sure that you connect the right connector to the right plug (there is notated which one is CRMA58 and which one CRMA59).
Make sure that CRMA16 and 15 are also connected. To find them follow the cables from the plugs or simply search for CRMA16 and 15 on the door. They are on the bottom left side on the door.

FANUC

Exercise 8

# Different Types of Stop

FANUC

Exercise 8

# Different Types of Stop

## Table of contents

## Background

Stops are one of the most obvious functions a robot or any machine must have. But understanding the different stops and why we use different stops is not so obvious.
A robot that has finished its work has to stop.
A robot that is doing a motion that it is not supposed to do has to be stopped.
There are lot of different scenarios where we need stops and we are going to give you an overview over those different stops.

## Equipment

For this exercise, you need:

Education Cell

## Different Stop Categories

There are 3 different categories for stops (following IEC 60204-1):

- Category 2 stop

Category 2 stop is a controlled stop where after the stop, power is still left available to the actuators (in our case motors)

- Category 1 stop

Category 1 stop is a controlled stop, where after the stop, no more power is left available to the actuators.

- Category 0 stop

Immediate stop with power removed from the actuators. The motion path of this deceleration is uncontrolled.

You see that the different categories are more ore less severe. Note that the more severe a stop category is, the more harmful it is for robot and the more safe it is for a human or outside equipment. To recover from a category 1 or 0 stop, remove the cause for the stop (release E-stop button, close the fence…) and press the reset button. Then you can continue or restart the program (this is recommended after cat 0 stop).

For further information about stop categories, refer to OM section 7 of the Safety Precautions “Stop Type of Robot”.

## Emergency Stop (E-Stop)

Emergency stop is used in an emergency.
This stop is used if people are in danger or the robot is about to collide with outside equipment, objects or also other robots. It is primarily used to avoid damage or injury as a last option and is therefor a category 0 stop. This is the most brutal stop for the robot but also the safest as it stops the robot immediately.
Now try and use this emergency stop button while the robot is executing a program.
Program Advanced_Prog_Ex should be well suited for this purpose.
Start the program and then, while in motion press the Emergency stop button. Observe how the robot stops.

**PLEASE NOTE THAT**

In an emergency stop, the robot may leave it's original path. After reset and start, the robot will first move back onto its old track and then continue.

For more information about emergency stop, refer to OM procedure 6-1 "Emergency Stop and recovery".

## Deadman Switch Release

In order to jog the robot or execute a program in teach mode, the Deadman switch has to be held in the intermediate position.
This is done to prevent the robot from injuring the operator. In case that the robot approaches the operator fast, the operator can release the Deadman switch and the robot will stop, or the operator may panic and grab the Deadman switch harder, so then the robot will also stop.
From the description above you can understand that this has to be also a category 0 stop as the safety of the operator stands in first place.
Try and reproduce this stop by executing the Advanced_Prog_Ex program in teach mode while grabbing the Deadman switch and then release the switch and observe if you see the same behaviour than with the E-Stop.
Again, while continuing the program, pay attention as the robot is probably slightly besides its track and will do an unexpected movement right after start.

## Open Fence

Normally, the robot is surrounded by some sort of fence. This can be a physical fence, but also a light barrier where the robot detects when someone enters the operating zone. In our case, it is the door of the cell. In either way, when the gate is opened (or door) the robot has to be stopped. But in this case, the robot has some time to stop. The norm says that the robot has 1 second to stop as the operator needs some time to get into the reach of the robot.

This is the reason why opening the fence uses category 1 stop, as entering the operating zone does not mean direct danger for that person.

Try and run the program again and this time, while operation, open the door and try and see the difference between the two category 0 stops before and the category 1 stop now.

Because this is a controlled stop, there is no danger to continue the cycle of the robot as it is still on the motion path.

## Hold

Hold is used to stop the robot in a controlled way in order to change something in the settings of the robot or the configuration of the robot. If an operator wants to stop the robot when there is no danger situation, he should always use the hold function.
If Hold button is pressed, the robot does a controlled stop but there will be no error message and the robot can be restarted from the position that it has been hold.
Hold is a category 2 stop.
Try this also with our program and continue (start) the program again.

For further information about hold stop, refer to OM procedure 6-2 “Hold and recovery”.

## Recapitulation

In this exercise, you should have learned the different stop categories available and in which case which type of stop is applied.
You should know the consequences from the different stops on people and on the robot.
You should be able to remove the cause of a stop, reset the robot and restart the robot.

FANUC

Exercise 9

# DCS Safe Zone

FANUC

Exercise 9

# DCS Safe Zone

## Table of contents

## Background

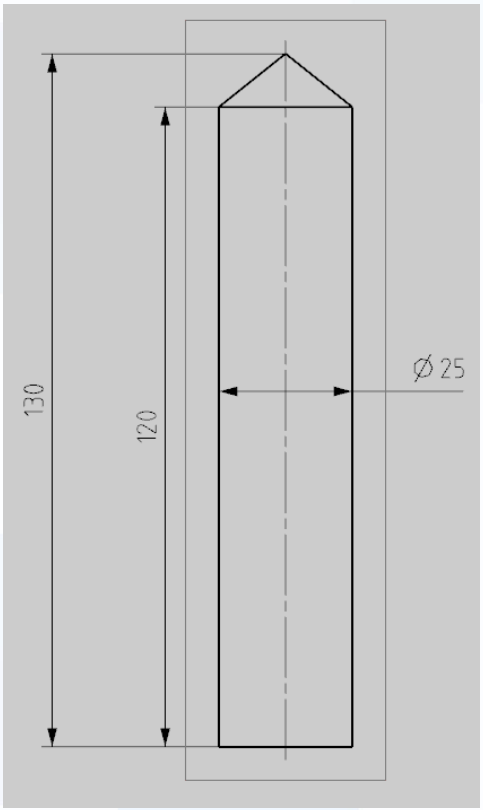
DCS Safe Zones are mostly used in industry when a robot moves in an area where other robots also operate or even people may get into the range of the robot.
In our case, the DCS Safe Zone makes sure, the robot can't get through the walls of the cell. As the name says it already, the DCS Safe Zone tells the robot in which zone it is safe to operate and stops it if it is about to get through the zone.
A zone can be set up from many different, simple zones combined, and so create more complex zones.

## Equipment

For this exercise you need:

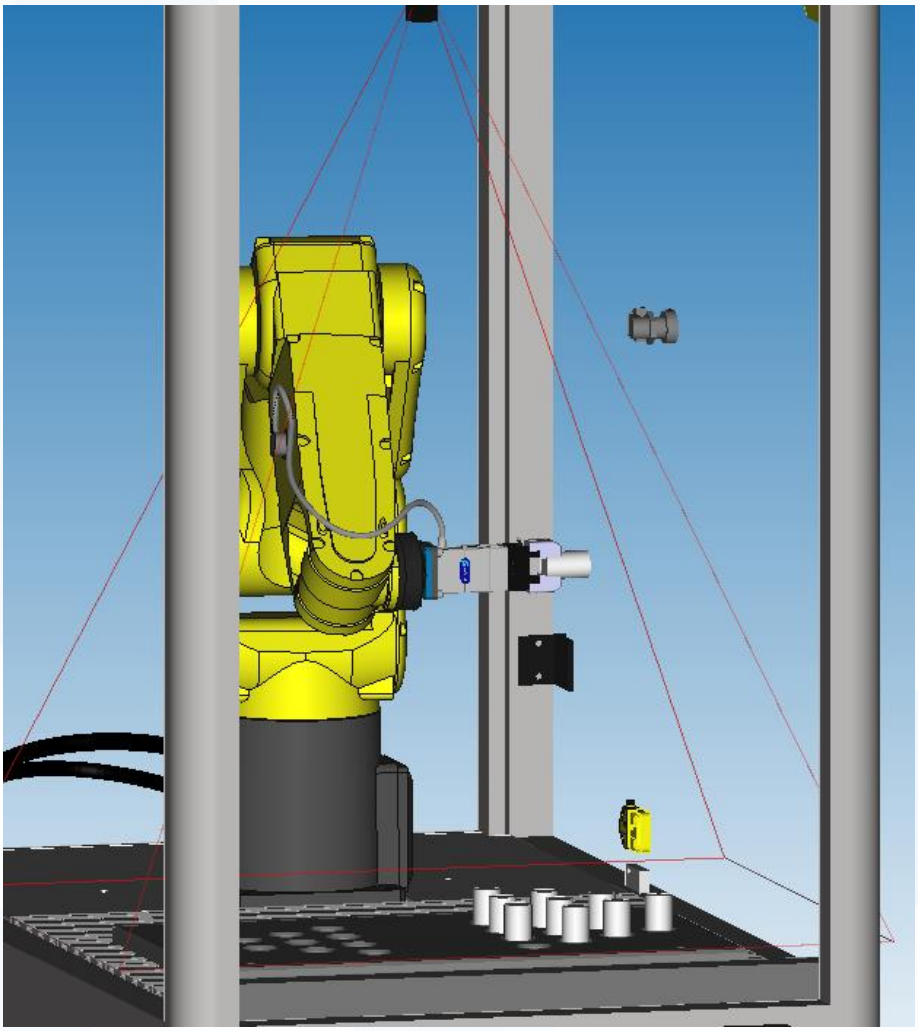
Education Cell

Fixed Pin – to be held in the gripper this time

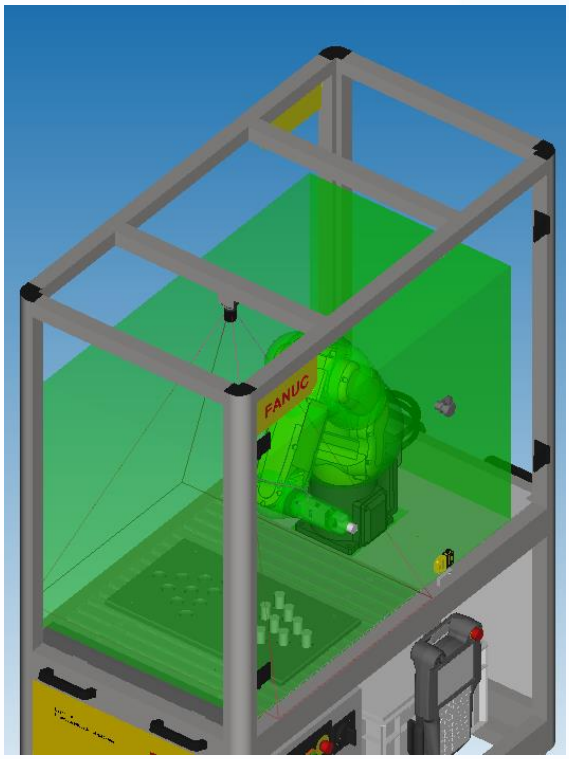
## Visualizing the DCS Safe Zone

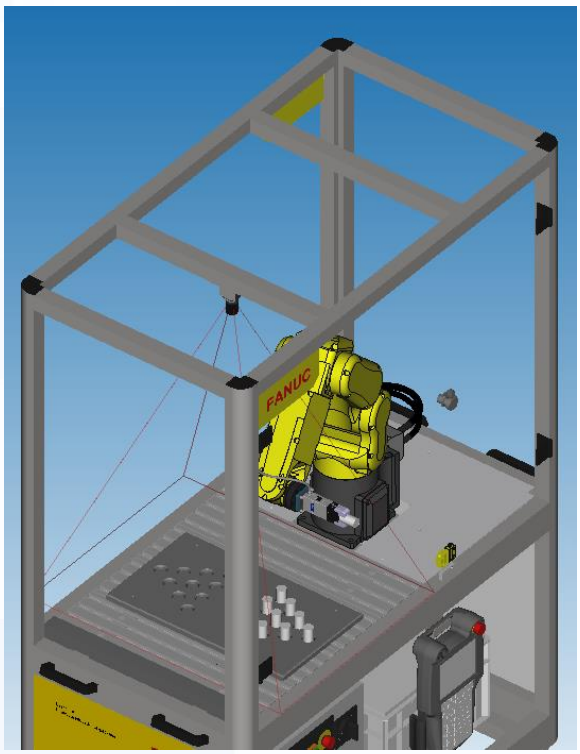
To visualize you the effects of the DCS Safe zone, we want you to try and move the robot out of the safe zone.
Turn the gripper by 90 degrees so that the tip of the gripper points towards the door.



## Visualizing the DCS Safe Zone

Jog the robot in direction of the door and watch how far the robot will move. At a given moment the robot will stop and it is not possible anymore to move the robot in any direction except away from the border of the DCS Safe Zone.



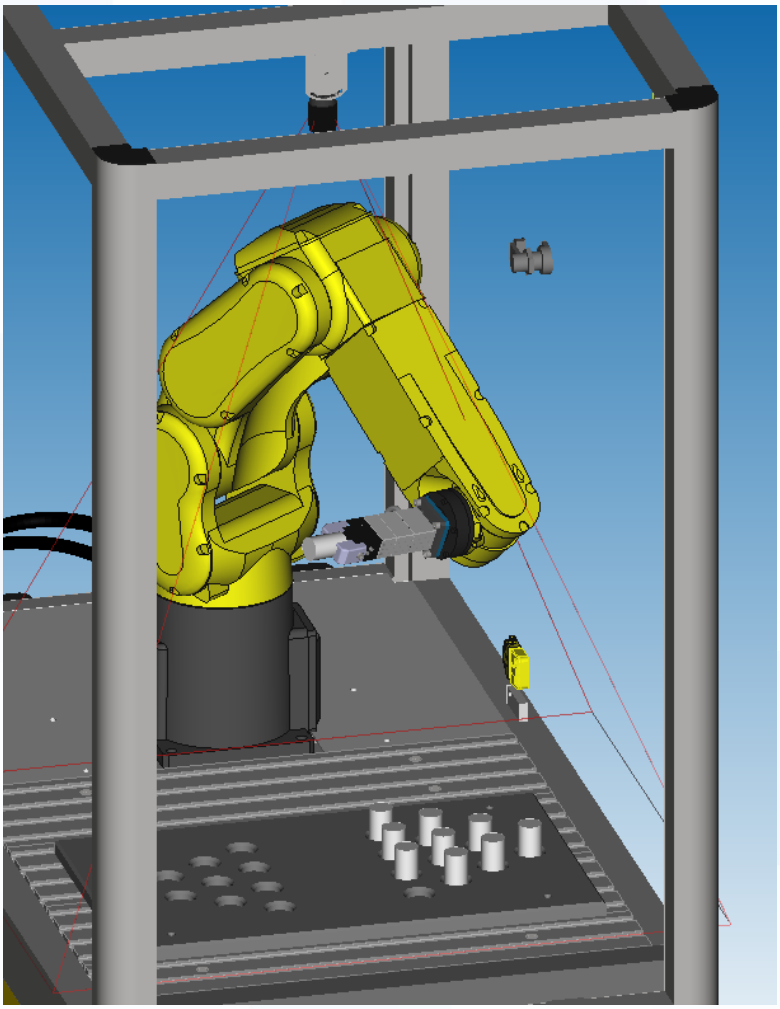




The TP will output the following error message.

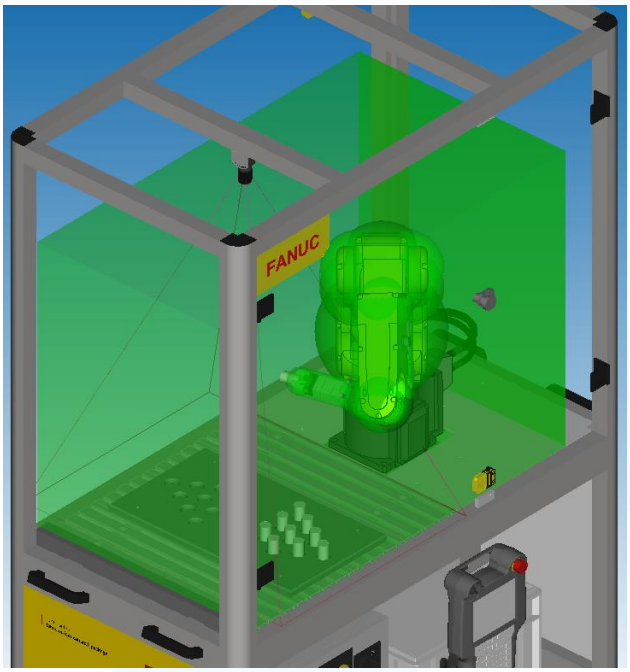

To release the robot from its error state, release the shift button, then press the shift button again and press reset. Now jog the robot away from the border of the zone.

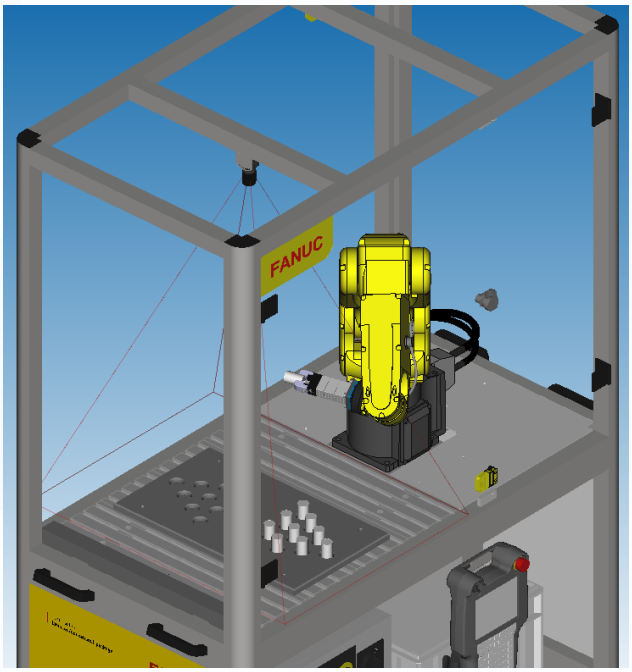
## Visualizing the DCS Safe Zone

Jog the robot away from the door and turn the tool so that the robot arm is near the door.
Jog the robot again in direction of the door. The robot will again stop in front of the door.



## Visualizing the DCS Safe Zone

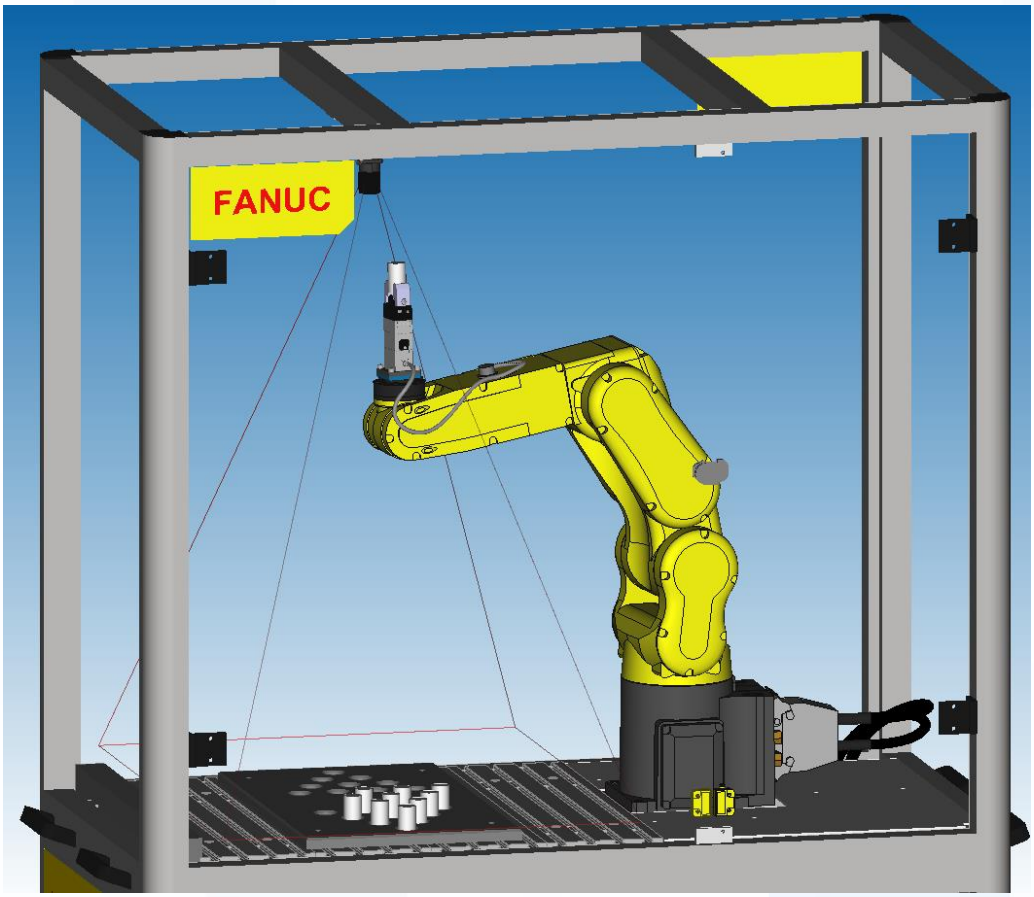



This is possible because not only does the controller verifies whether the tool is outside the box, but it also verifies the robot itself, so that no part of the robot can get into a dangerous position.

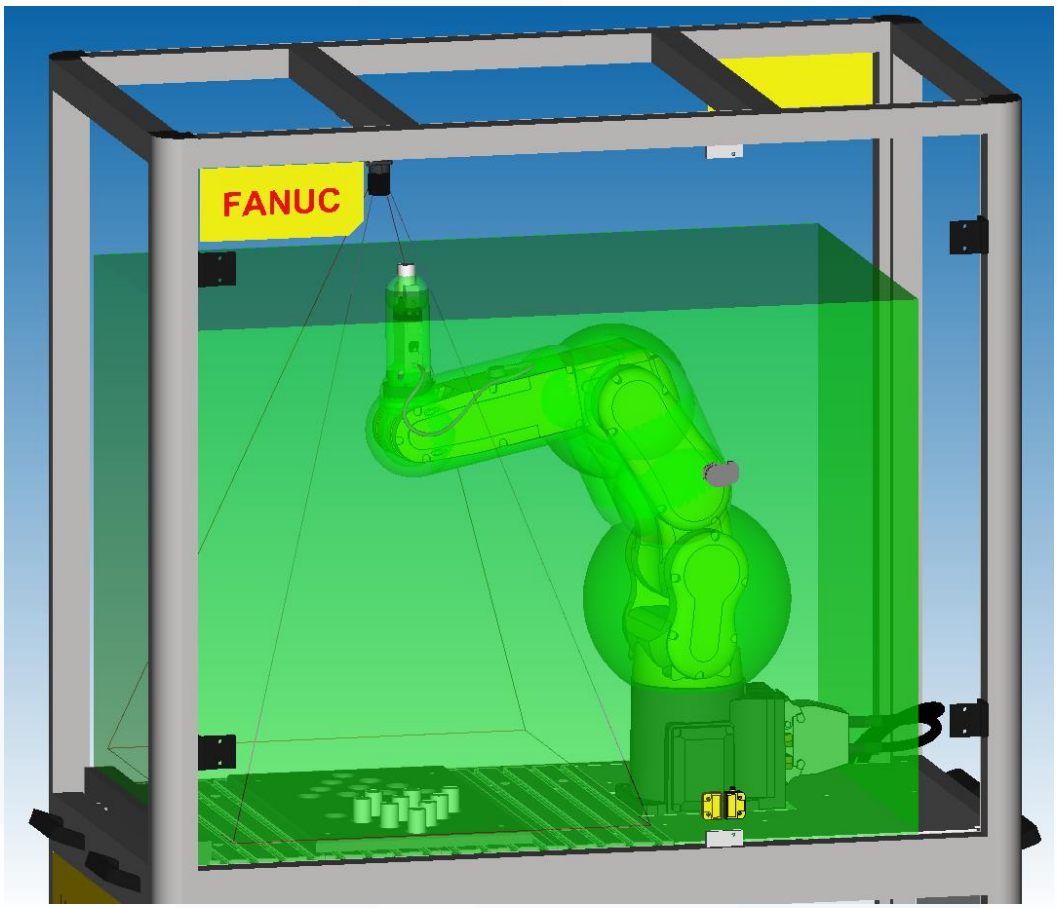
## Visualizing the DCS Safe Zone

Now we want to show you that not only the safe zone is on the side but also on the top.
In order to see this, jog your robot so that your tool is facing upwards.



## Visualizing the DCS Safe Zone

Then, jog the robot upwards until it stops again with the error message.




As next step, we will change the DCS Safe Zone and see the difference in the restricting area of the robot.

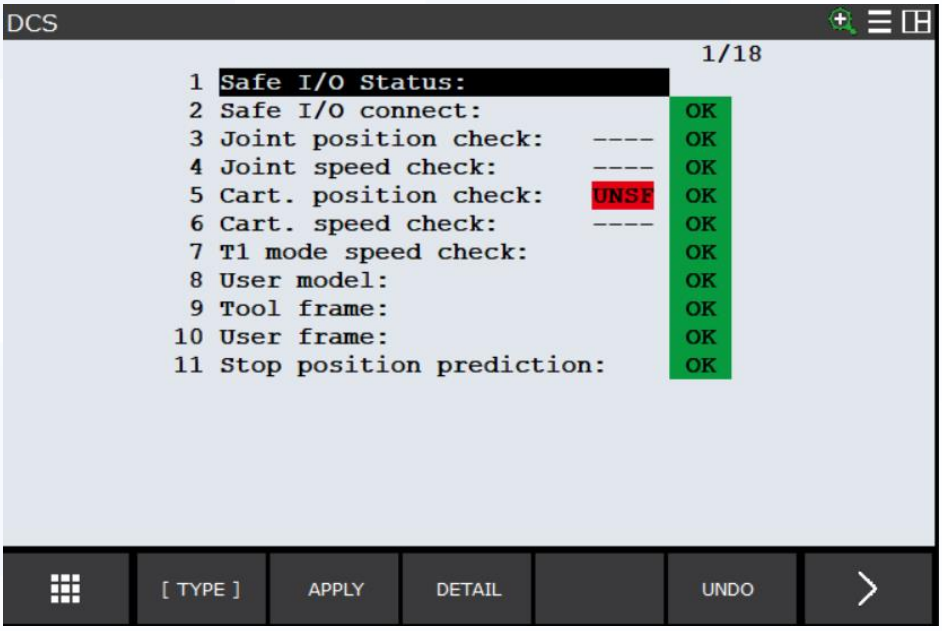
## Setup of DCS Safe Zone

First we need to get to the settings of the DCS. Therefore press the menu key, go to second page system (6) and then to DCS (8).

**PLEASE NOTE THAT**

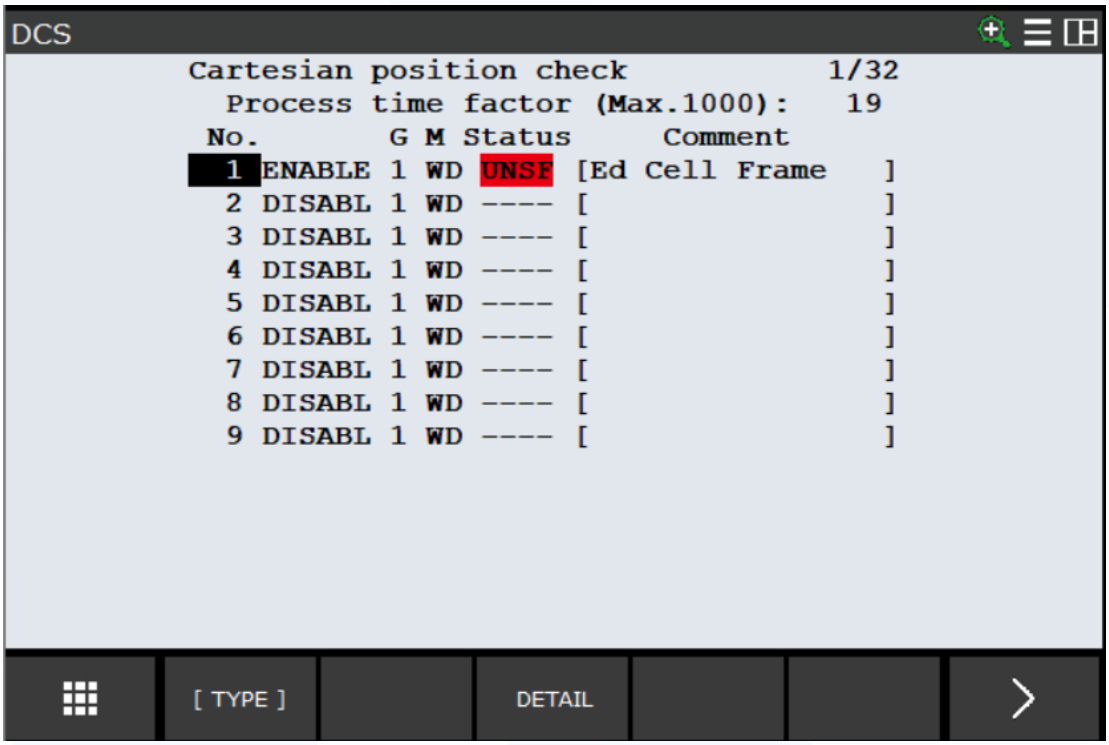
You are now about to change system files and settings. Make sure that you do everything exactly as we advise you to do, because otherwise serious damage and injury might occur!

There are different methods of setting a DCS Safe Zone. We are using the Cartesian position check. If you have not yet moved the robot, there should be standing UNSF meaning unsafe, because the robot is on the border of the Safe Zone and therefore not safe.



## Setup of DCS Safe Zone

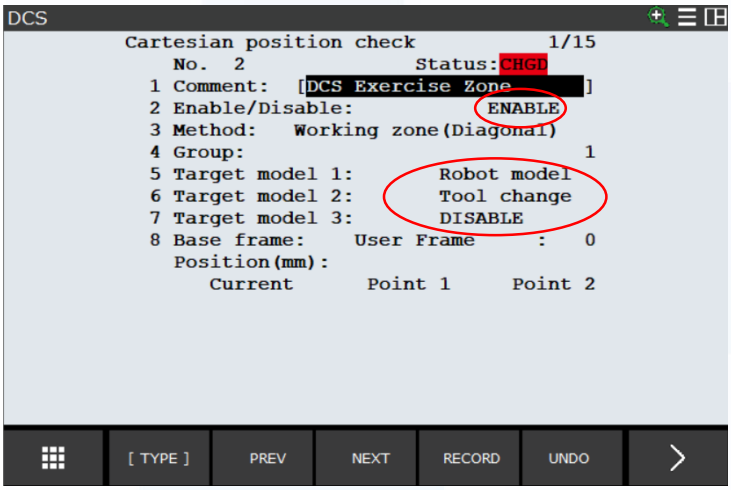
Go into the Cartesian position check.

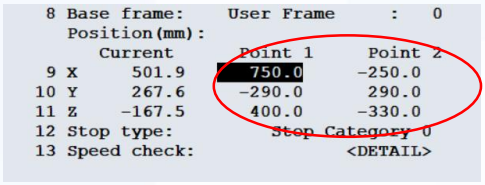



Currently there is only one Safe Zone enabled. But it is possible to enable more than just one Safe Zones and then the controller will constantly check if all the robot parts are inside (or outside depending on the individual setting) the enabled zones.

## Setup of DCS Safe Zone

We now want to create our own zone. Go to the second position and enter. Firstly you need to acknowledge that you want to make changes.

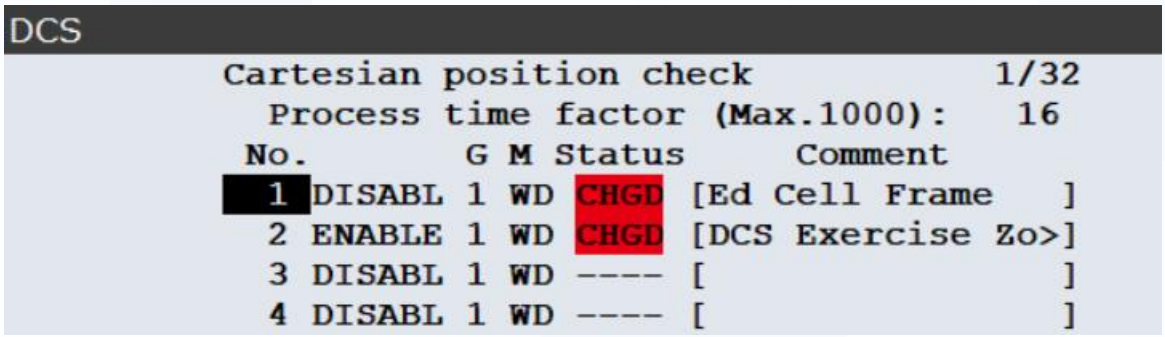



Then give our frame a name, set Method (3) to Diagonal(IN),Target Model 1(5) to Robot Model, Target Model 2(6) to Tool Change, and copy the position data for points 1 and 2.
Finally Enable this Frame (2).

## Setup of DCS Safe Zone

Now the our new Frame is enabled, but the old one is also still enabled. Go to the old one “Ed Cell Frame” and disable that Frame. You should now see that the old Frame is disabled, and the new is enabled.

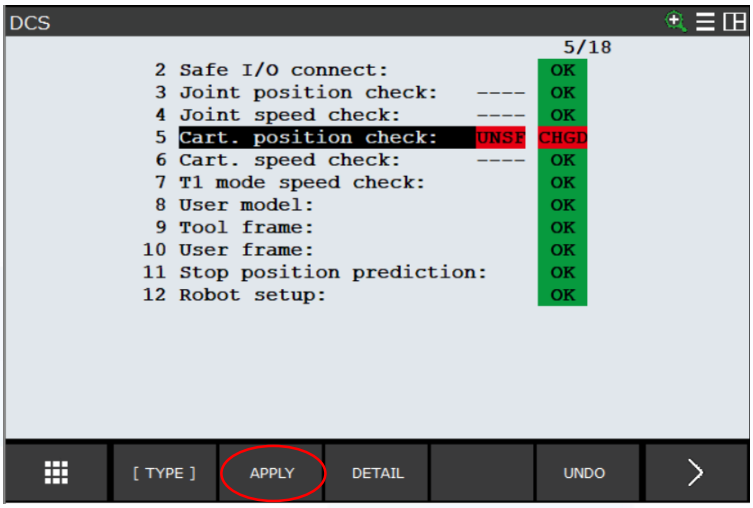



The CHGD mark means that you have to apply the changes and cycle power to make current settings active.

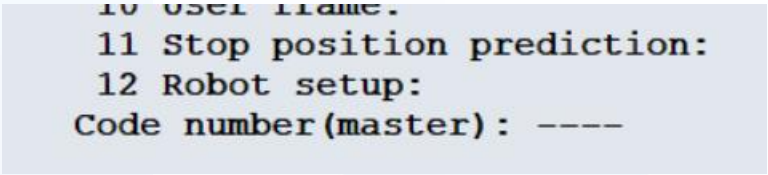
Those settings are not active until you applied and cycled the power!

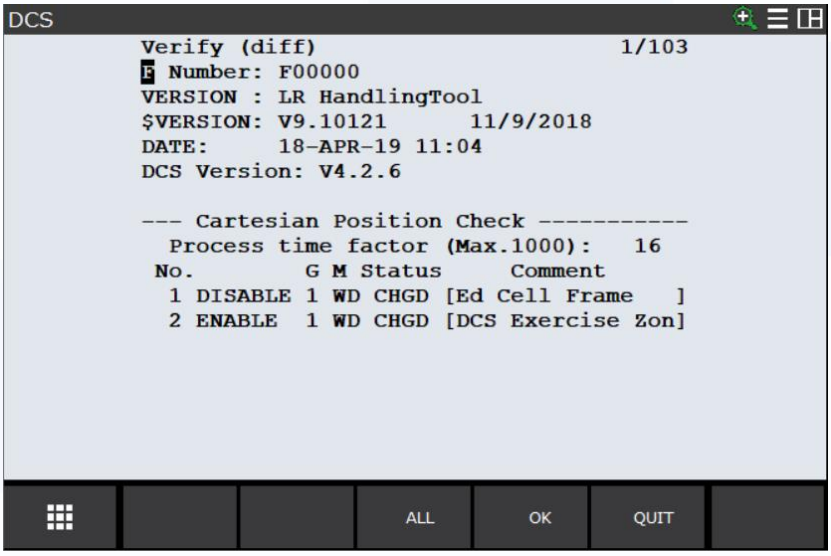
To make the change valid, you have to press the Prev button to get back in the first DCS menu. Press F2 (Apply).

## Setup of DCS Safe Zone



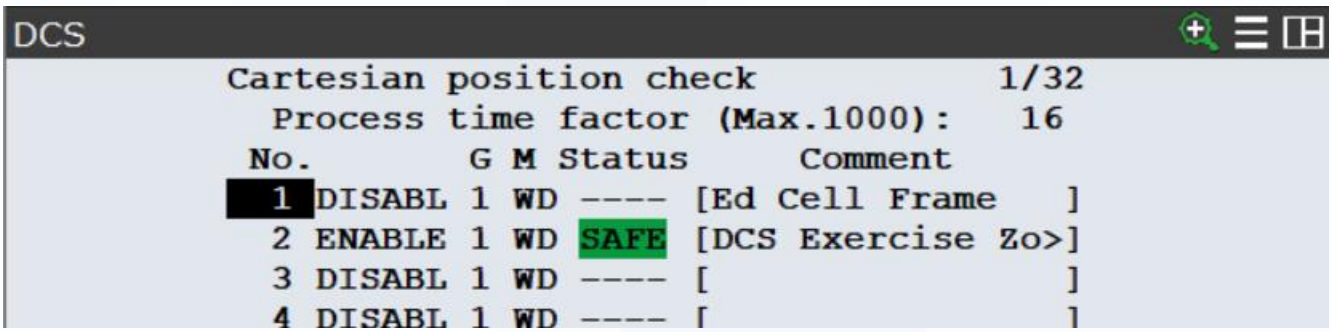

You will be asked “Code Number”, then enter 1111 then there will be a second window where you press F4 (OK). Now you have to cycle power. After this procedure, your new DCS Safe Zone is active.



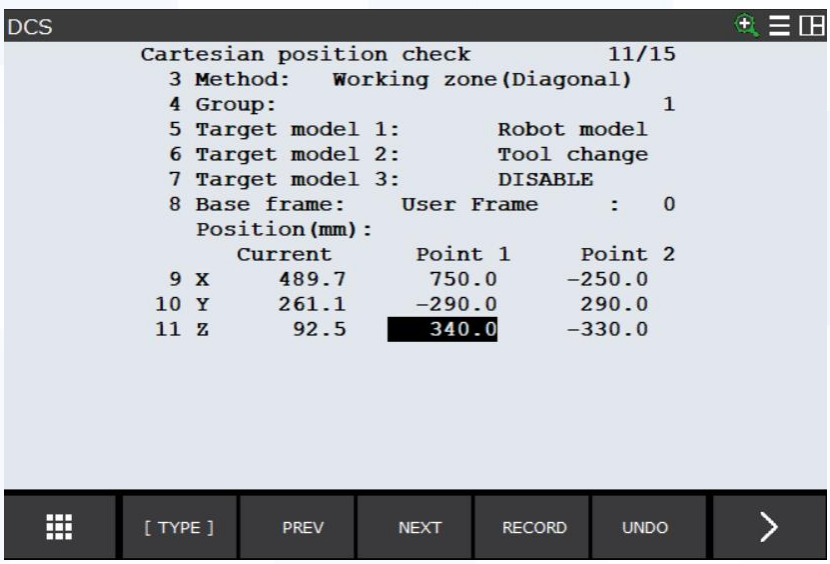



## Moving in new Safe Zone

The settings should now look like this. If it doesn't, go back to page 13 and do the procedure again.

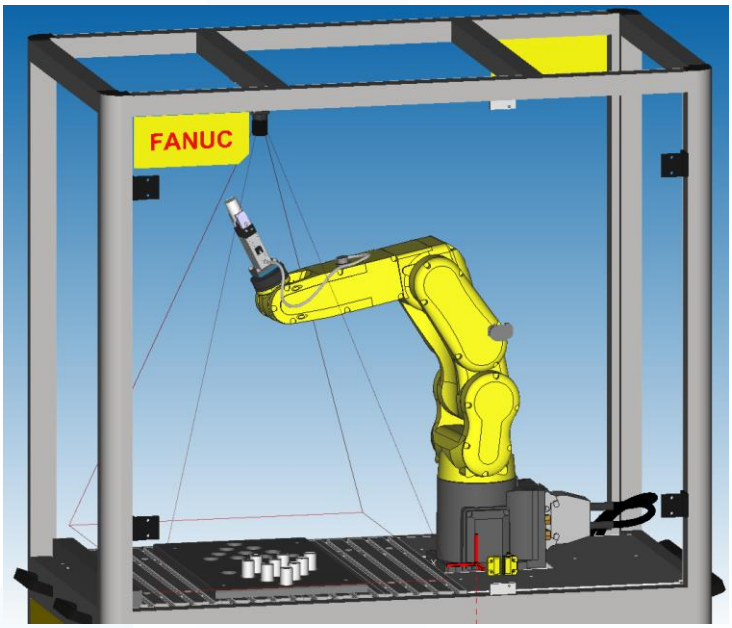



Now release the error and jog the robot a bit down (about 100mm).
We want to change the height of our frame, so go back into the settings.
Change the height from the point 1 (Z-coordinate) to 340. Apply the settings and cycle power.

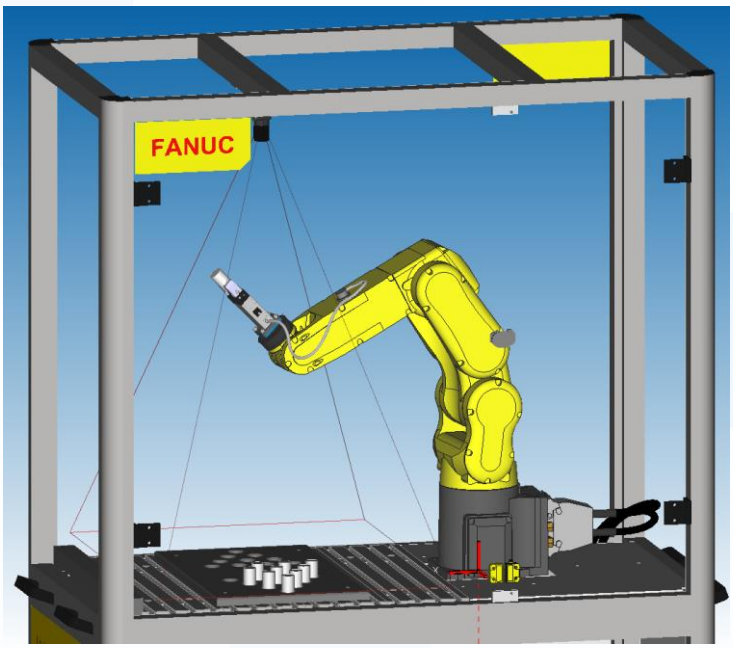


## Moving in new DCS Safe Zone

Now try and move up again. You will notice that the robot stops much earlier than it did before.



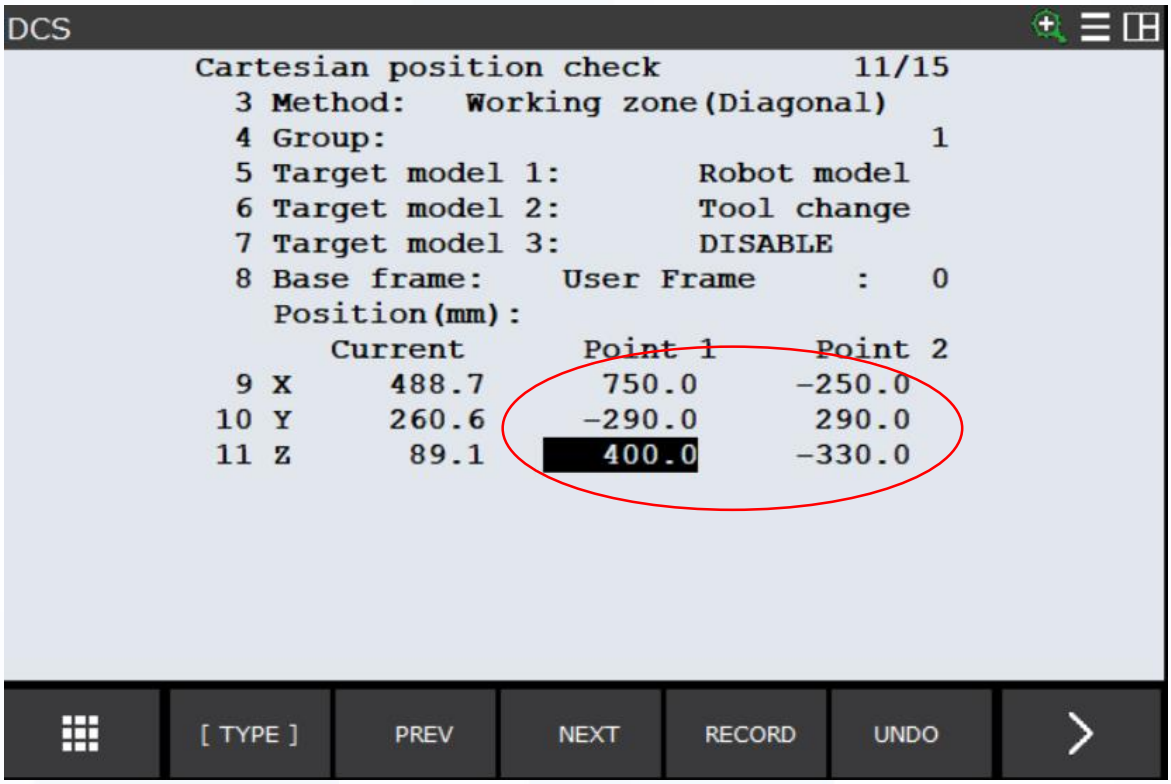

Before




After

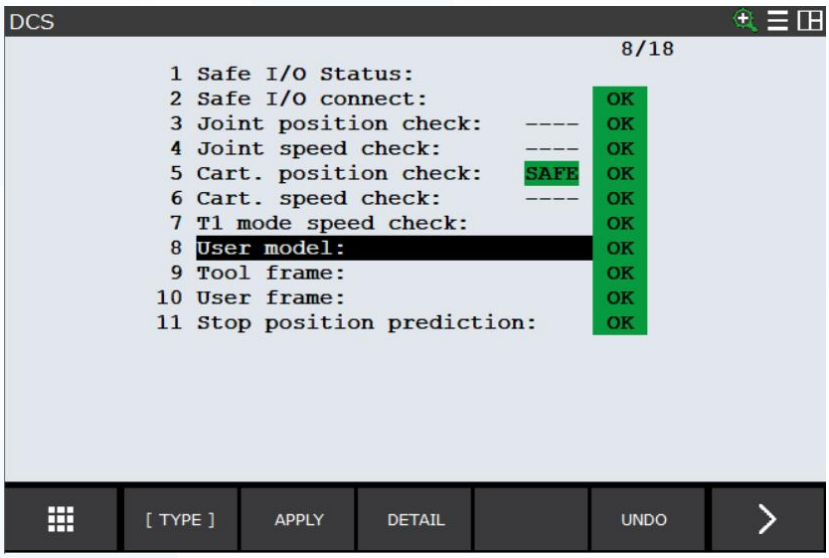
## Moving in new DCS Safe Zone

Now reset the Safe Zone to the old value (400). See the following picture to make sure the DCS Safe Zone is right.

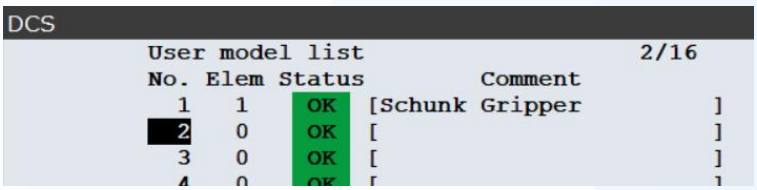


## Setup of User Model

The other possibility is to change the tool setting, instead of the zone setting. To do this, we firstly need to set the User Model. (There is another possibility, to use the Tool Frame, but we will only cover the User Model setup in this exercise)

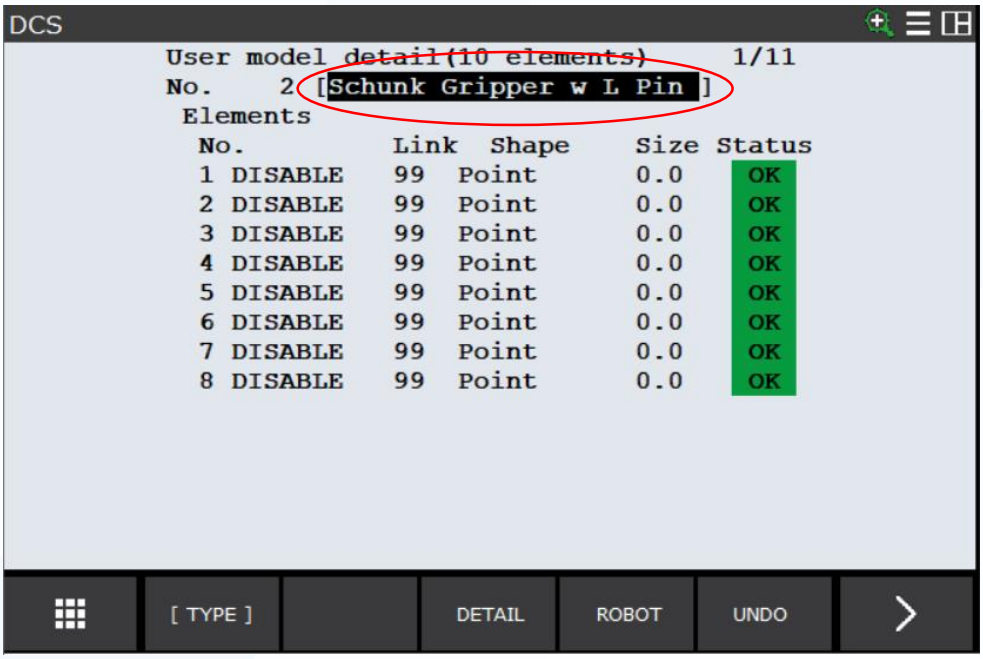



In the User Model settings there should already be the Schunk Gripper. We will however create a new one to show you how to do it.



## Setup of User Model

Enter the second one.
Firstly set a name.




We then have the possibility to set 10 different models, that, combined are used as our model. But for this simple tool, we only need one model. Enter the first model.

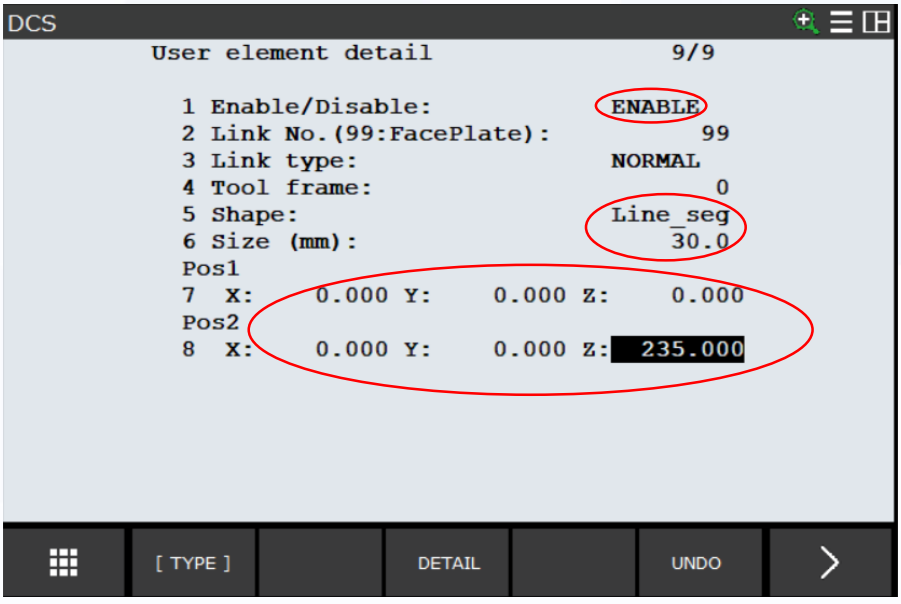
## Setup of User Model

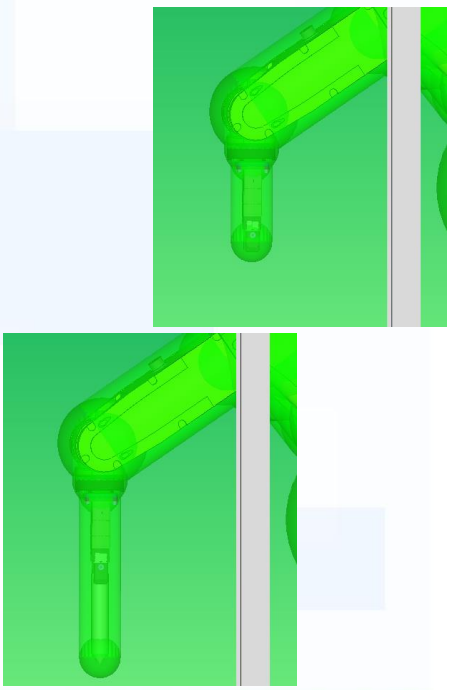
We want to setup our gripper with the long TCP Pin, we used for TCP teaching (table pin). As this Pin is considerably longer than the gripper, we need to teach a new model to our controller in order to avoid the tip of the pin to go outside the safe zone.

First of all we need to enable the current model, then we choose the Line_seg as shape, insert 30mm as size and then change the Z position of the 2nd point to 235. (distance in mm from the final flange to the tip of the Pin)

For more information about the different possibilities to set up a User Model refer to manual R-30iB_Plus_DCS_Manual section 3.4. “Setup of User Model” and 4.2 “Setup of User Model”.

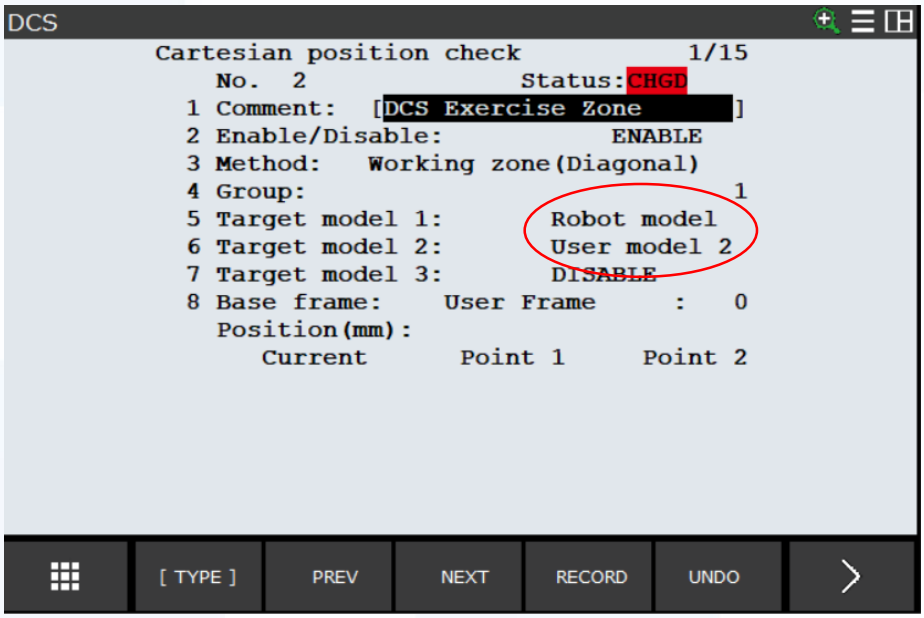
You can set this distance according to the length of your TCP pin, if it is shorter or longer. The length from the flange to the tip of the gripper is 115mm.



## Setup of User Model

Now as we have setup the gripper model, we need to apply this model in our cart. position check. In those settings, we have the possibility to set 3 target models, which the controller is checking while operating. The first target model (Robot model) is the robot. This means that the robot itself cannot exit the safe zone. As target Model 2 we will set user model 2, the tool we just created.




Now you have to apply again the settings and cycle power. Before you cycle power, jog the robot down, so that the gripper will not end up, being already beyond the zone border.
Your new Tool should now be setup. (Make sure to actually put in the long TCP Pin in the Gripper)

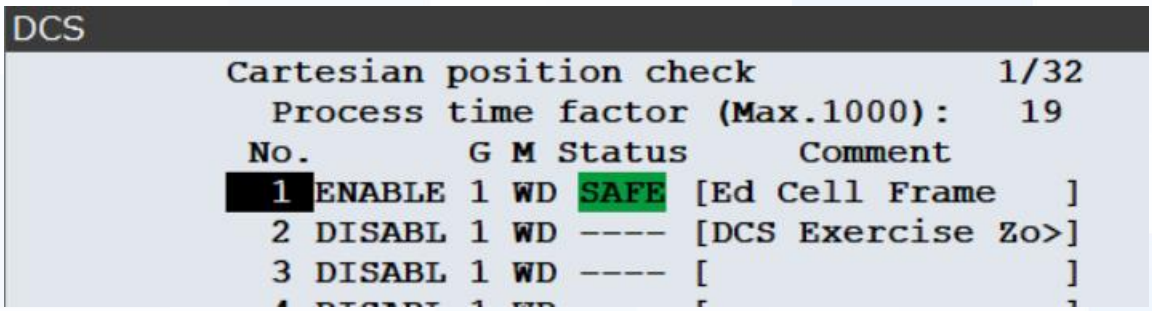
## Setup of User Model

To check, jog the robot up until it stops. If you have done everything right, the TCP Pin tip should stop where the gripper tip stopped the first time.
You can try the same on the side wall.
If you want, you can change between the “Ed Cell Frame” and your Frame to see the difference between the two settings.

**PLEASE NOTE THAT**

You need to make sure that the TCP Pin is out of the gripper, because otherwise, the tip can crash into the walls!

In order to do this, make sure that you enable the first zone, then disable the second zone, apply the settings and cycle power.
After you are done, make sure to make the first zone active again (enable the first one, disable the second one) so that no dysfunction might occur, due to bad safe zone settings. Apply the settings and cycle power. Make sure the settings are accepted until you finish this exercise.



## Recapitulation

In this exercise, you should have learned what a DCS Safe Zone is and how it operates and works.
You should be able to set up a safe zone on your own and also set up a User Model according to your tool that is attached to your gripper.
You should know how to apply the settings.

FANUC

Exercise 10

# Customized *i*Pendant Screen

FANUC

Exercise 10

# Customized iPendant Screen

## Table of contents

## Background

User defined HTML pages are used to give the operator a visual feedback of what the robot is doing.
It can also be an interface for the operator to give orders to the robot without the need of additional inputs (refer to I/O Box).
You have already used some HTML pages while operating the robot without specifically knowing it. The pages displayed during the AAA_Demo program were actually customized HTML pages called by the program.
As you may remember, those pages helped to give a the possibility for the user to input some commands and to get some input about what is happening with the robot.

**PLEASE NOTE THAT**

In this exercise, all references are made to iPendant_customization_operator_manual_[B-83594EN_01] (iCOM), except if announced differently.

## Equipment

For this exercise you need:

- Education Cell
- A PC with Roboguide installed
- A Memory Stick

## MS SharePoint Designer Installation

We recommend you to do the .stm sites on the Microsoft Office SharePoint Designer 2007. This is a Software free to download.
A detailed description of how to install SharePoint to your PC refer to iCOM section 3.3 “Installation”.
Make sure you add ActiveX Control to your toolbar, you will use this afterwards for every command you add to your page.(refer to iCOM section 3.3.1 “ActiveX Control Shortcut”)

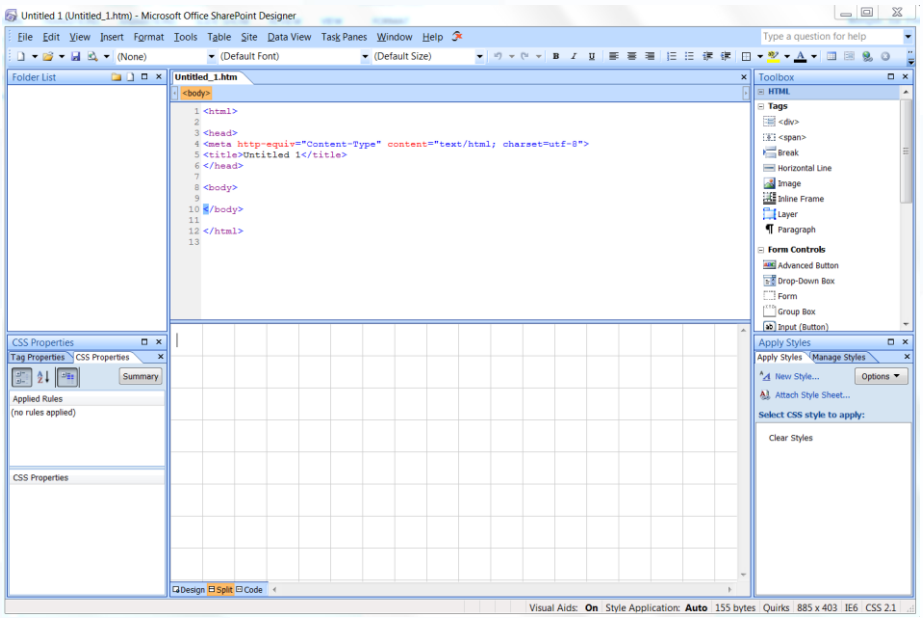
## First Steps

In this exercise, we will not cover all the different controls available. You should however be able after this exercise, to use those controls not covered in this exercise with the help of those covered and the manual.
To get a quick overview of all the controls available refer to iCOM section 3.4 “Control Features Summary”.
The following controls will be covered by this exercise:

- Label
- ToggleLamp
- ToggleButton
- Image (not in manual)
- Multi

## First Steps

Now we will start our exercise, by creating a new document in SharePoint.

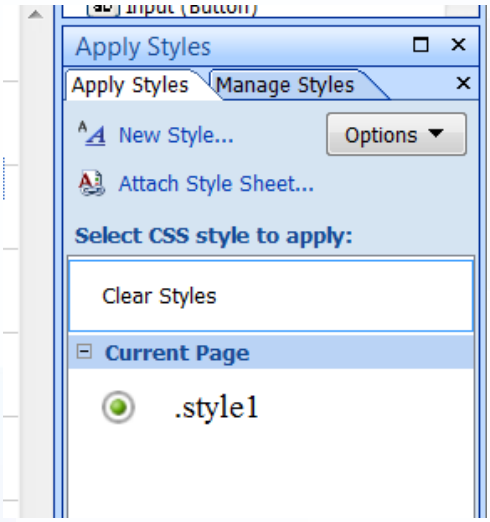



Firstly we want to change the title of our page to Exercise10 first page.

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Exercise10 first page</title>
</head>

<body>

</body>

</html>

```

## Label

We start by adding a simple label control. Click the ActiveX Control icon in the toolbar, select the FANUC iPendant Label Control in the list and press OK.
You have now a label on your page. You can position that label with space, tab and enter.  Alternatively you can input a new style on the right bottom.




You will be able to set position of the label, font settings etc.
To apply those settings to a label (or other control) change the <p> to <p class="*nameofyourstyle*">.

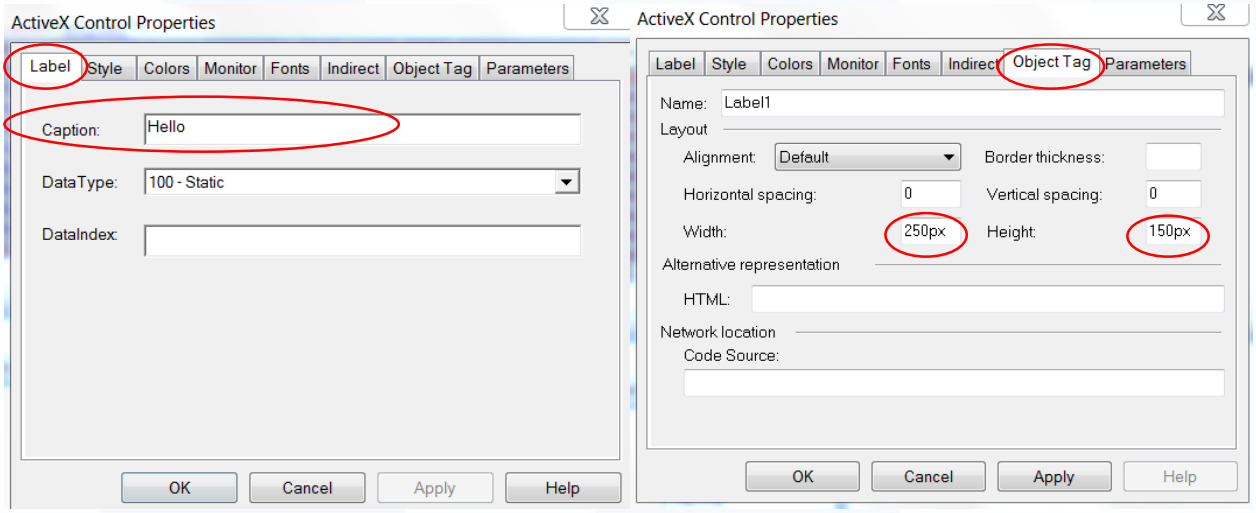
```
20 <p>
21  </p>
22 <p class="style1">
23 <object classid="clsid:710
24     <param name="Caption"
```

## Label

To change the settings of the control, double click on the control.

For detailed information, refer to iCOM section 5.4.1 “Label Control”.

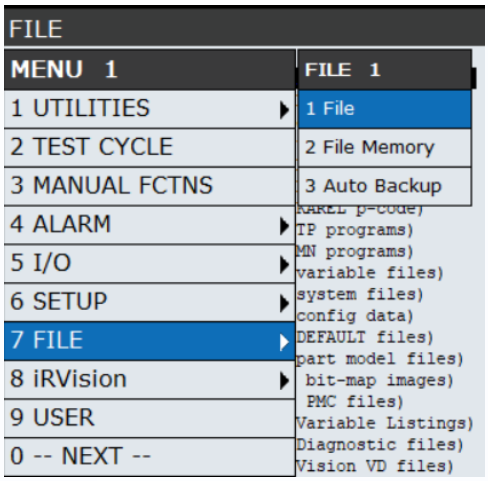
Further in the exercise, we will go a bit deeper into several settings. For now, add a Caption in the Label settings and change the size of the box in the Object Tag settings to 250px width and 150px height.

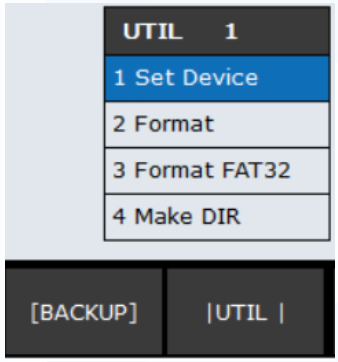


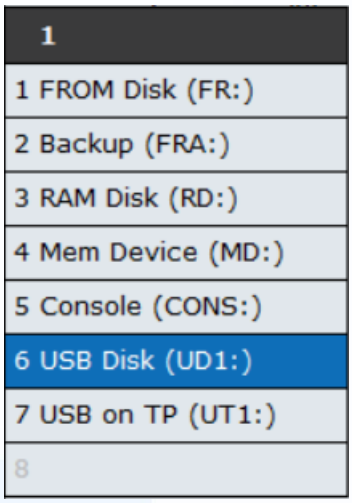

Now save your document as an .stm (simply add .stm behind the name of your document).

## Upload the Page to the Robot

To upload the page to Roboguide, you need to put the document in the UD1 folder from the robot. This folder is located at *Documents\My Workcells\FEC_Education_Cell_V2_4\Robot_1\UD1*.
To upload the page to the real robot, you need to put the document on a USB MS and plug it into the controller.
The following procedures are the same on the robot than on Roboguide.
Press the menu button, then go to File (7). Press F5 (Util) and Set Device. Choose USB Disk (6).

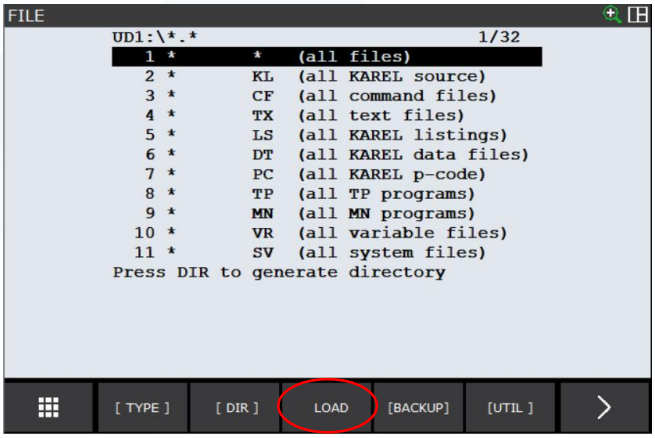


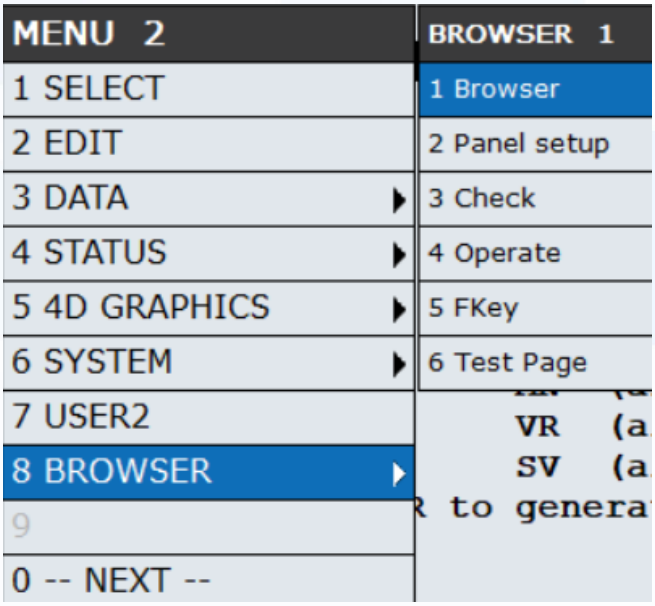



## Upload the Page to the Robot

As you have only the .stm file on the USB MS, you can press F3 (Load) and load all files.
You now have the page on the robot controller.

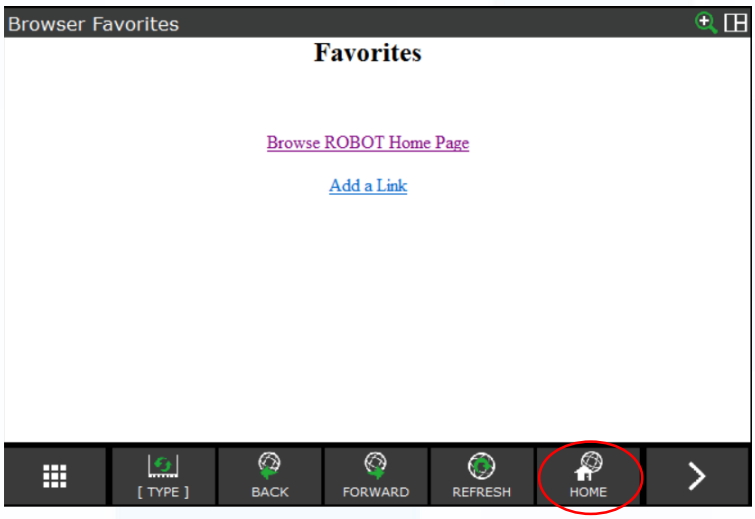


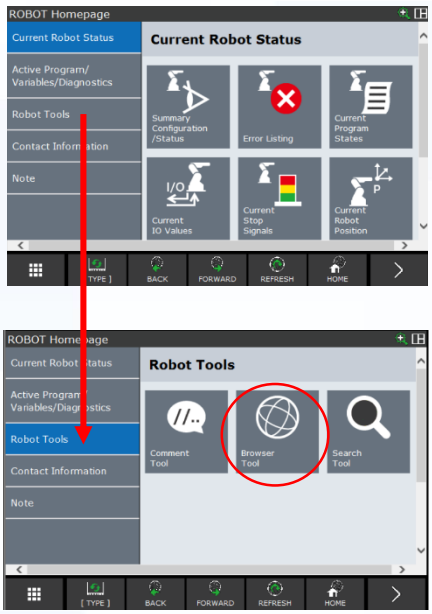

The only thing that now has to be done is to link the page. In order to do this, press the menu button again, then go to the second side and select browser (8).

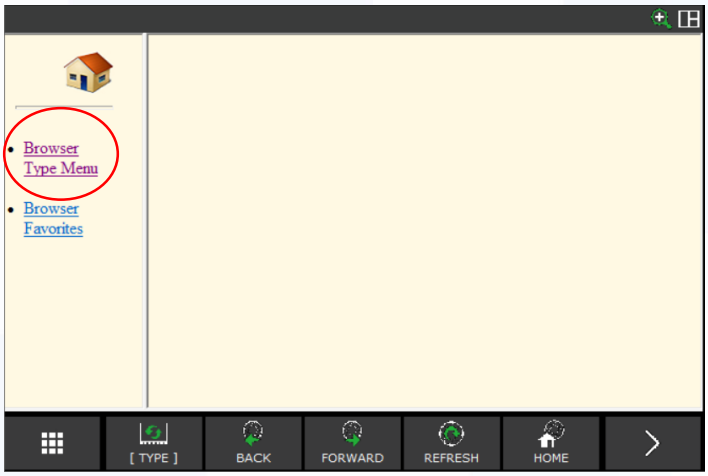


## Upload the Page to the Robot

Press F5 (Home) and go to Robot Tools and select Browser Tool. Then select Browser Type Menu and add your page to the list. The address will be /fr/*nameofyourdocument*.stm.



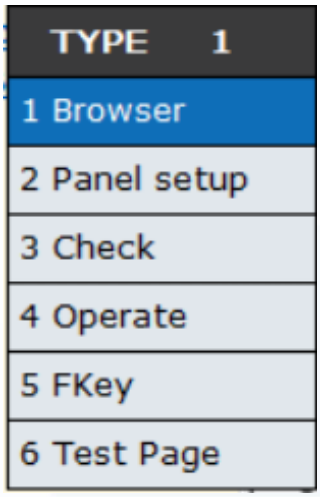



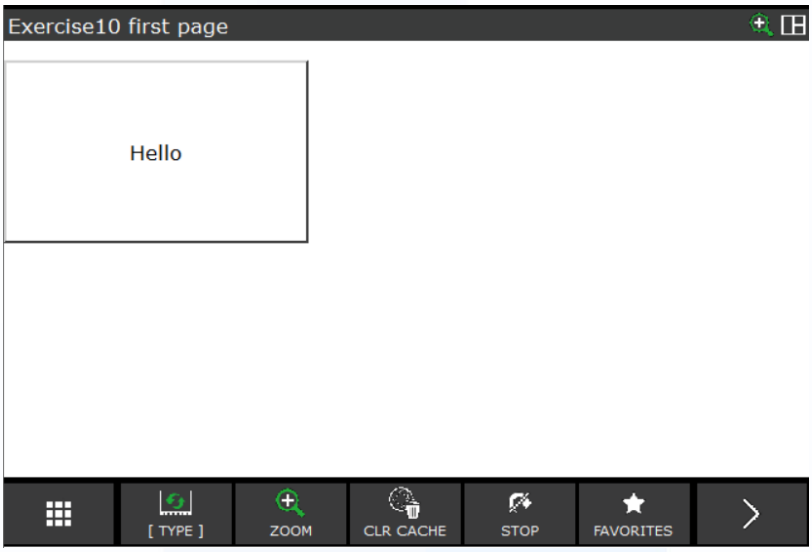




| Browser Menu | Name | Add |
|---|---|---|
| [1] | Panel setup | ipnl/pnlgen.htm |
| [2] | Check | /fr/EdCellChk.stn |
| [3] | Operate | /fr/EdCellOp.stm |
| [4] | Fkey | /fr/EdCellFkey.st |
| [5] | Test Page | /fr/Ex10.stm |
| [6] | | |

## Displaying the Page

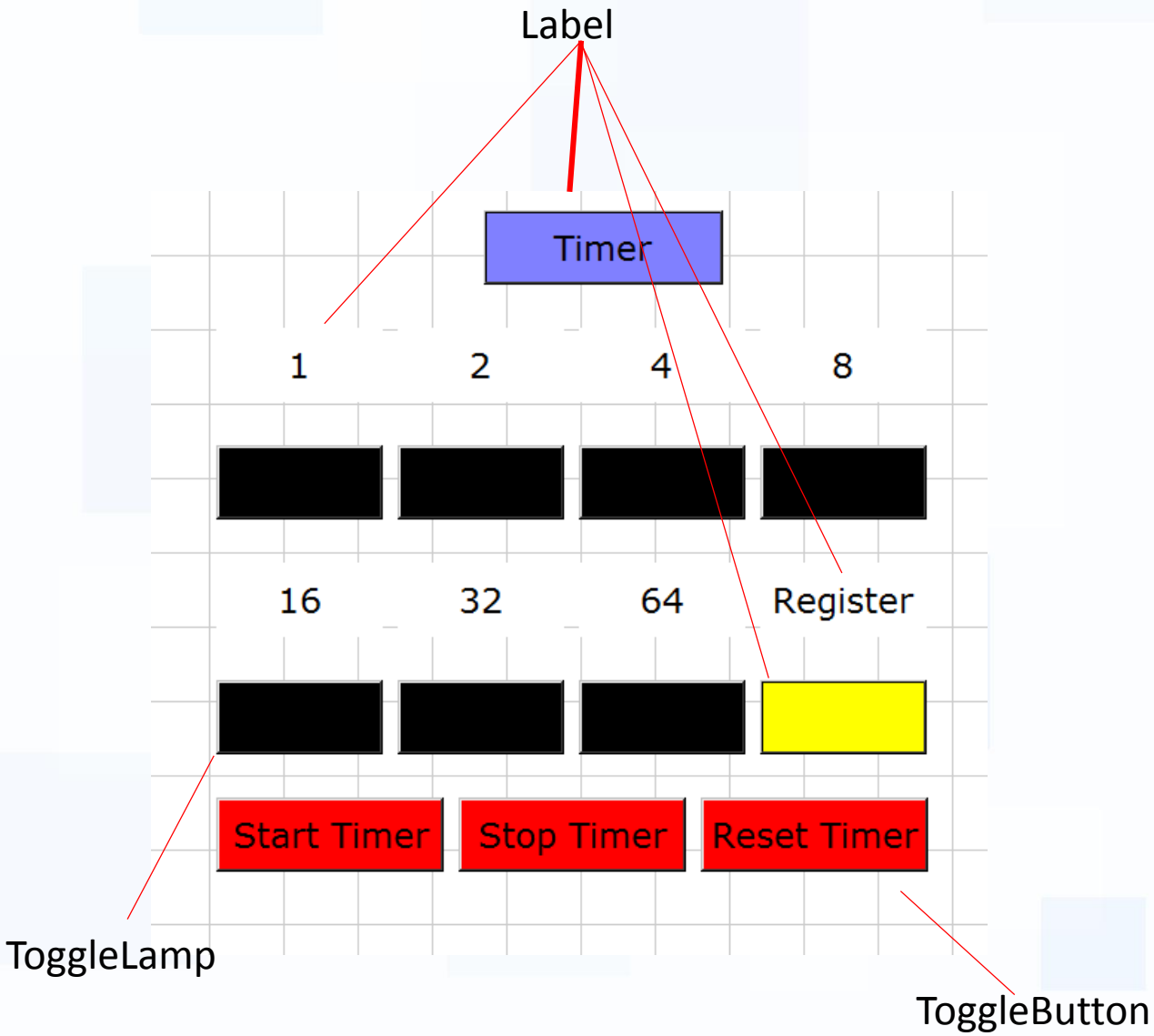
If you have done everything as told, your page should now be linked in the robot controller.
To display your page press F1 (Type) and select the name you assigned to your page. You will get to your new page.
Make sure that after each change of your page or new upload to press F3 (CLR Cache) because otherwise it may be that the actual new page is not (correctly) displayed.



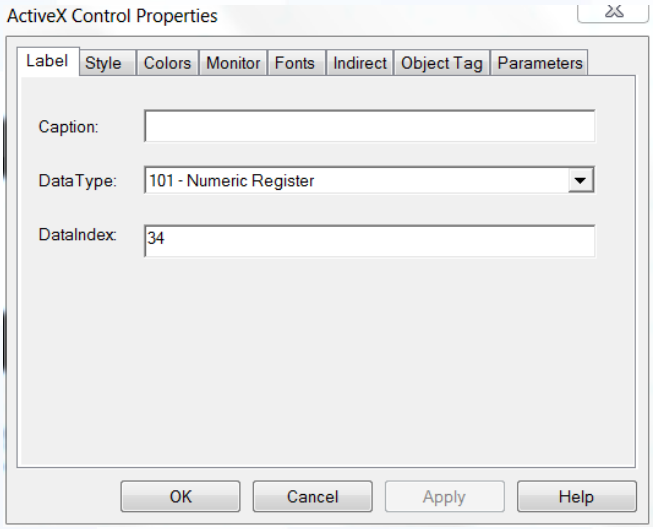
## Buttons and Lamps

Now that we know the basics about programming, uploading and displaying a page, we want to create a User Interface for the Stopwatch program. (refer to exercise 7)
We need to have 7 lamps to represent the outputs and 3 buttons to represent the inputs. And finally a Label box to display the register.
The design of this page is up to you. You should end up with something similar to the following:



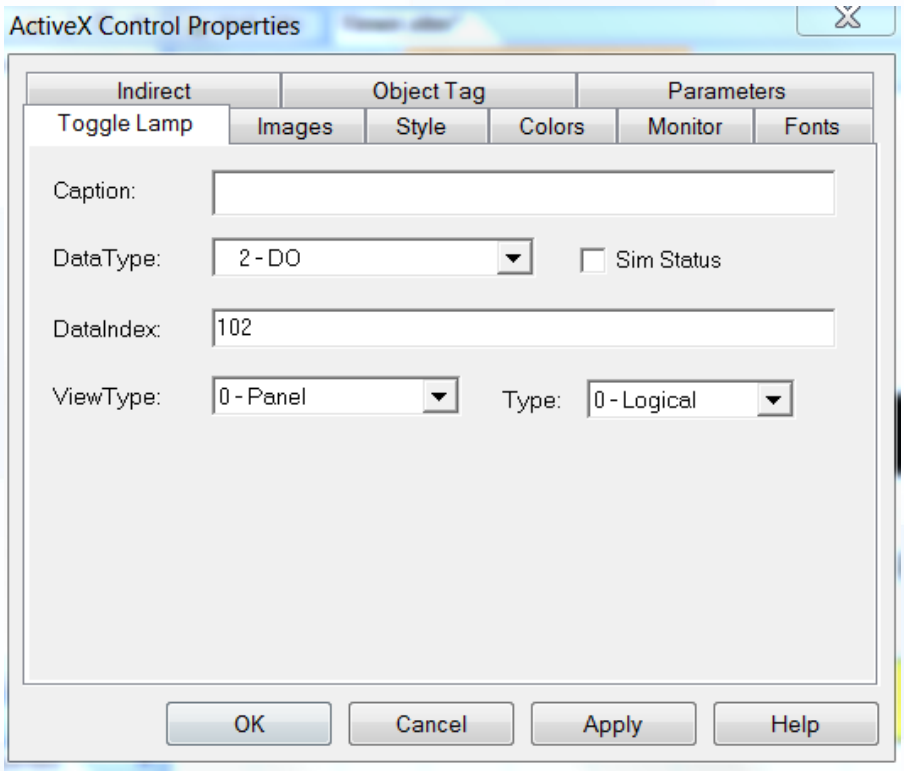
## Buttons and Lamps

All labels except the yellow one (you can give it the colour you like) can already be set as in the previous page.
As for the yellow label, you need to change the data type to 101-Numeric Register and then input Data Index 34. That should be the Timer register if you haven't changed anything. Check the register if it still is this value and otherwise change the register in the label.



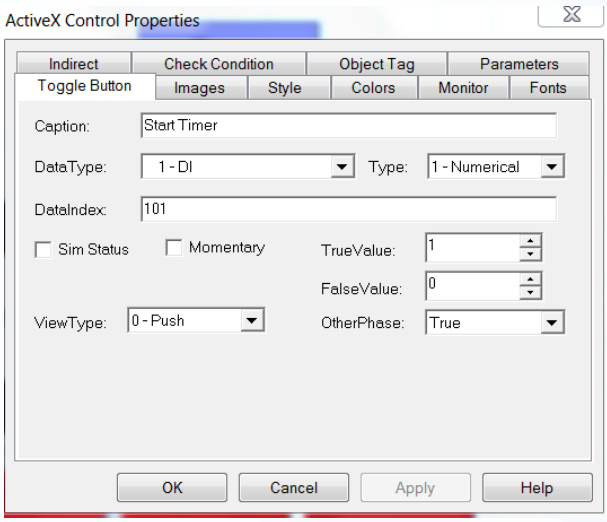
## Buttons and Lamps

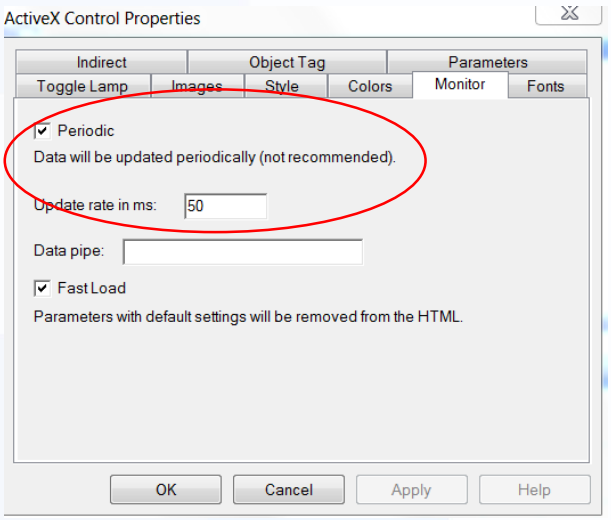
The lamps and the buttons work quite similarly to the label we just set up.
For the lamps, you need to switch the Data Type under ToggleLamp to 2 - DO and input the DataIndex that corresponds to the lamp. (102 for the first lamp, 103 for the second…for explanation refer to exercise 7)
Under Colors, you can change the colours for true and false, as well as the colour if you want to have a caption.



## Buttons and Lamps

For the buttons, you need to switch the Data Type to 1 - DI under ToggleButton and the Data Index according to the buttons. (101 for first to 103 for third)
You should also put in captions for the buttons. (Start Timer, Stop Timer, Reset Timer)
You should also change the monitor settings. Check the Periodic Box and change the update rate to 50ms. This will avoid that the lights seem as if they are not really a clock, which we want them to be.

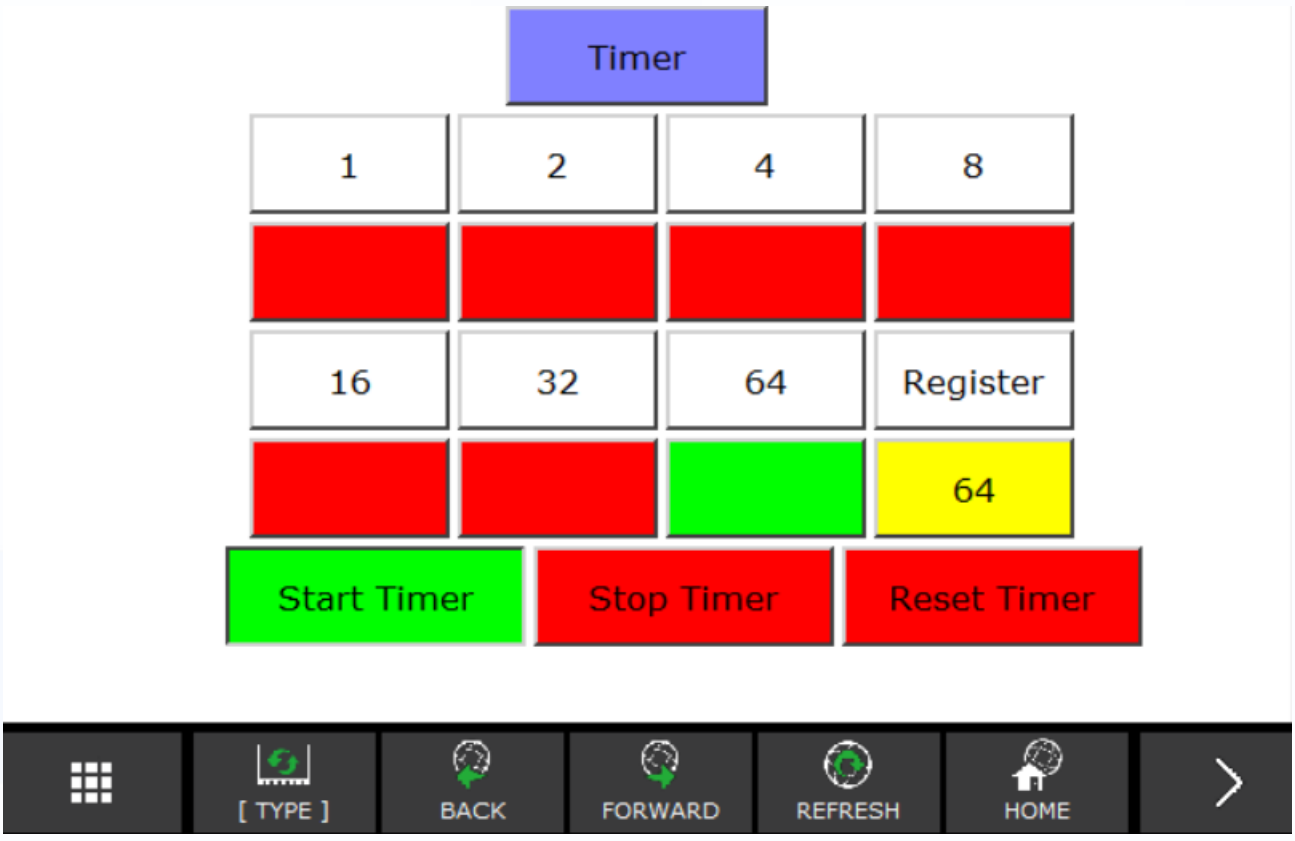


## Buttons and Lamps

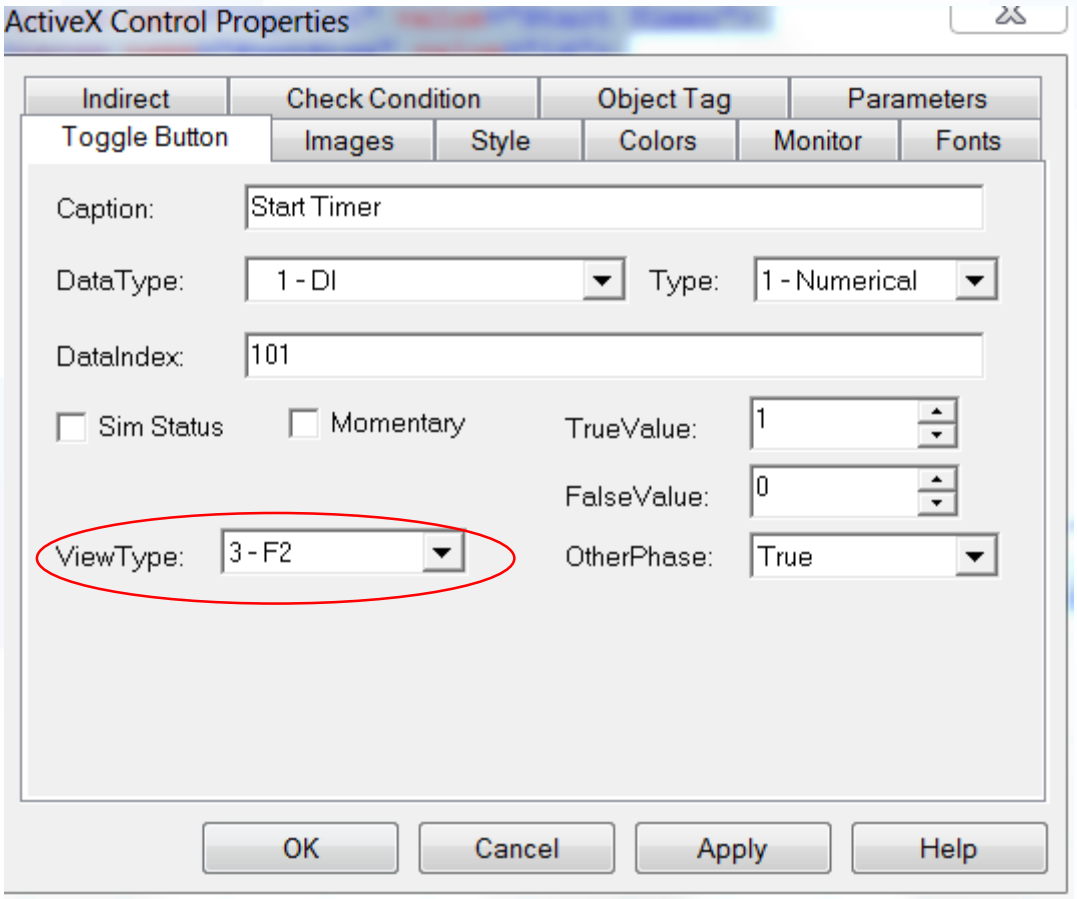
Save your page, upload it to the robot and link it.
Start the I_O_Timer program, then switch to the web page and check whether the inputs and outputs work. (togglelamp, togglebutton)
In order to this to work, you have to set the DI101 to 103 to Sim, because otherwise, the robot cannot change the input without an actual input.

For further Information about how to simulate DI refer to R-30iB_Basic_Operator_Manual section 6.4.2 “Simulated I/O” and procedure 6-11 “Simulated input/output”.



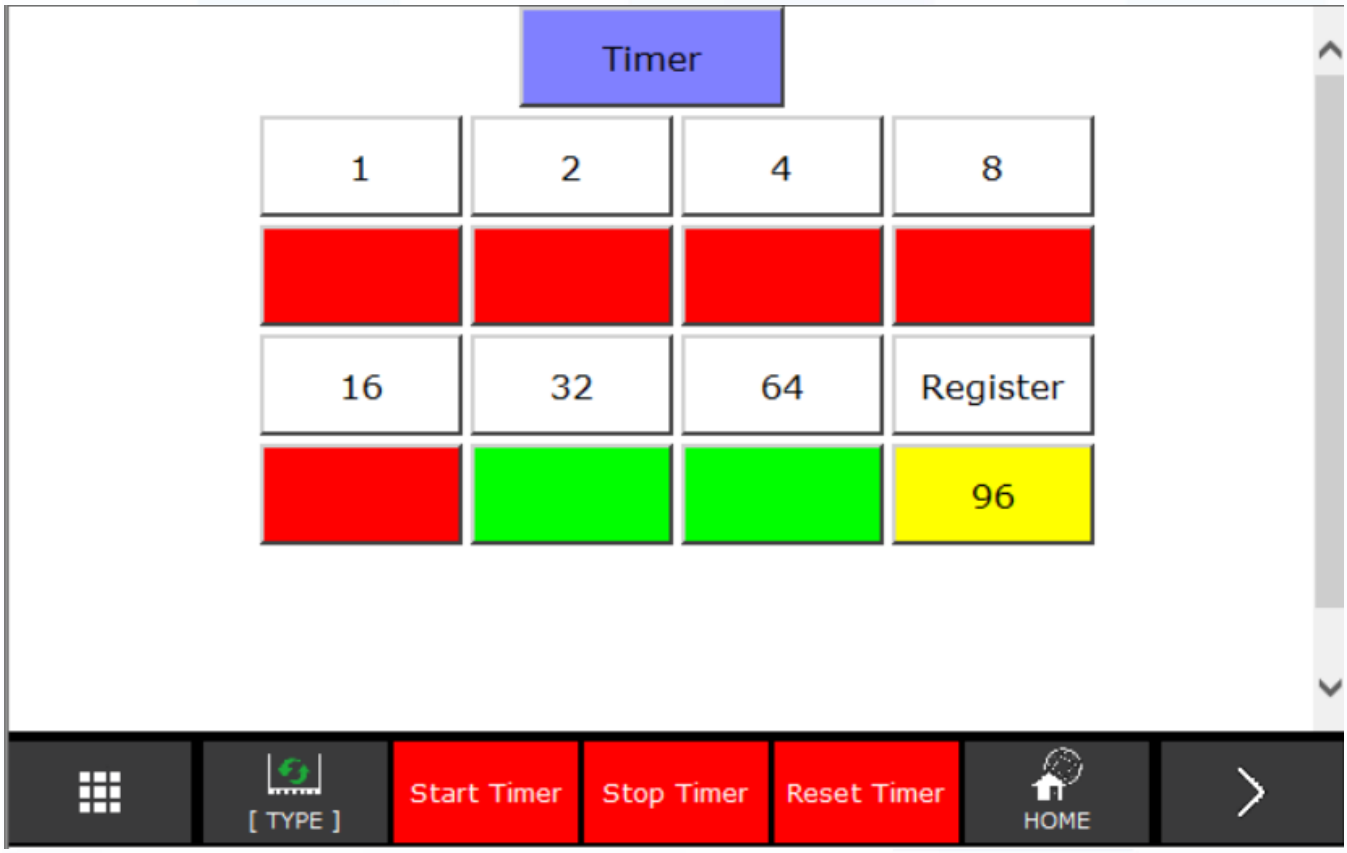
## Buttons and Lamps

To make the operation a bit easier, it is possible to link some buttons to the F2-F5 buttons.
To do this you need to go to the settings of the buttons and switch the View type from 0-Push to an F. (F2 for Start Timer for example etc.)



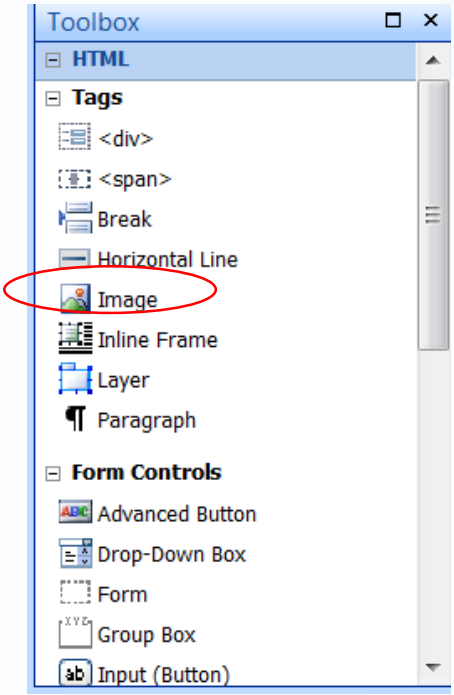
## Buttons and Lamps

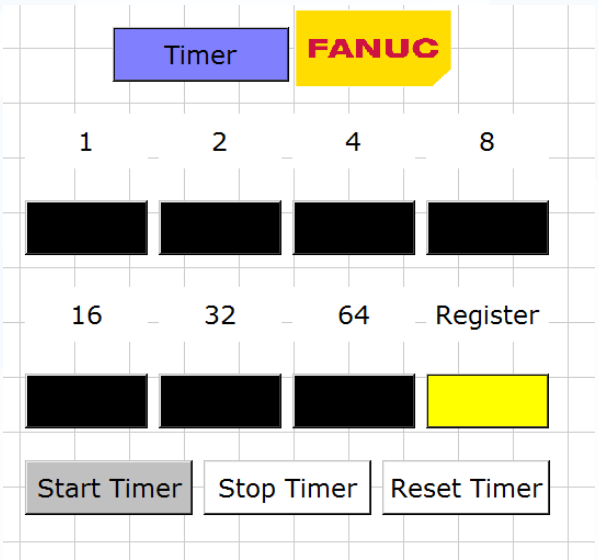
Save your page and upload it to the Controller and try the program again.
Your buttons should now be linked to your toolbar.



## Images

You also have the possibility to add pictures to your page.
The simple Image command, is NOT in the ActiveX Command list, but on the right side in your toolbox.




In the image settings you can browse for pictures and insert them.

**PLEASE NOTE THAT**

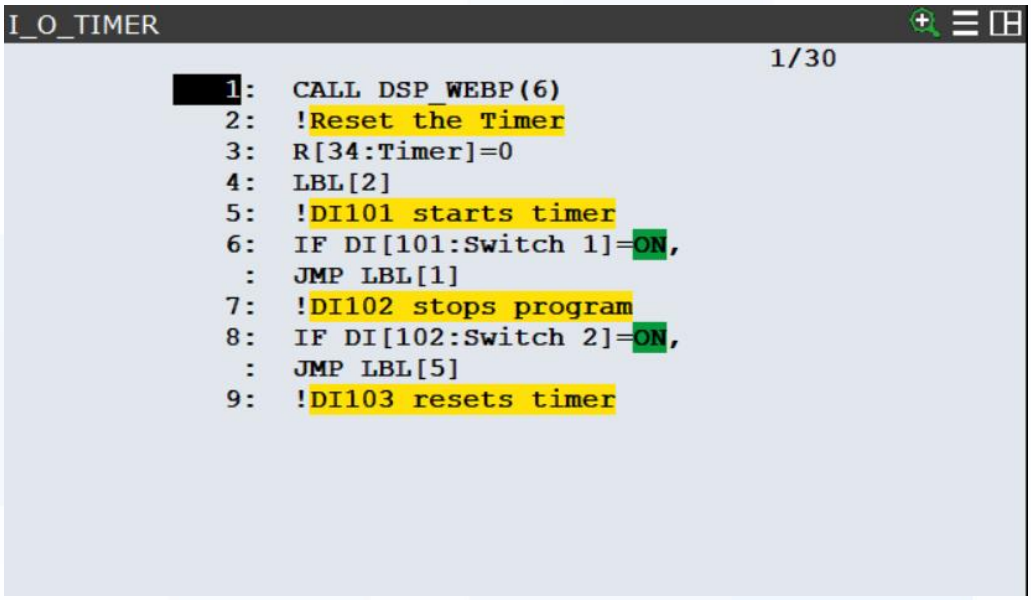
You should make sure to only input .img, .gif and .jpg pictures because the controller can't read the other formats.

## Images

To upload the page, the procedure is nearly the same than before, only that you need to put the picture also on the UD1 and upload both the page and the picture. (by uploading *all files, you do not need to worry about something not being uploaded)
Link the page again, clear the cache and then you should now have the picture in your page. (If you haven't changed the name of your page, you do not need to link the page again, just go to home and clear the cache, then go back and the new page should be displayed.

## Call a Page from a Program

Now as we have a nice interface, we want our program to call that interface as soon as it is started. There is already a program in the controller called DSP_WEBP() which will call a selected webpage. The number of your page is the number which is in front of your page in Type.
Yours should have number 6.
Add a line in front of the program and call DSP_WEBP(6).

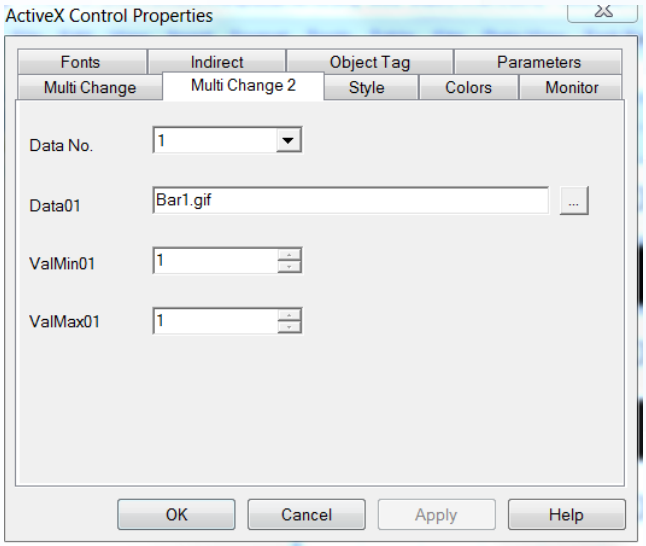


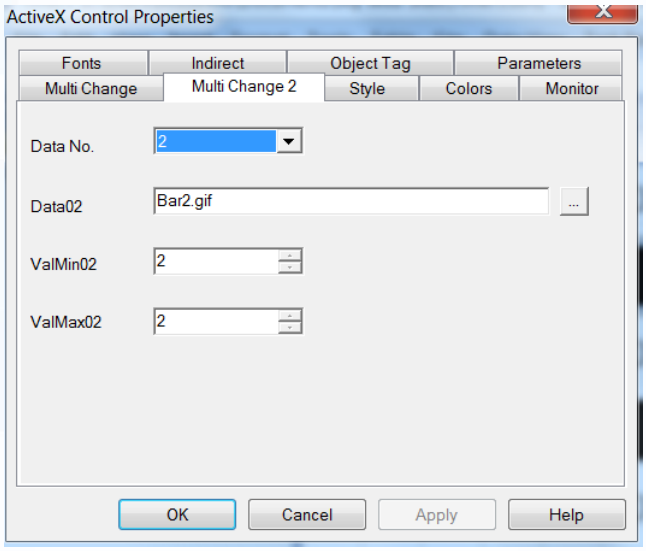
## Multi Control

This is the last function described in this exercise. The Multi Control can be considered as an image but which can change for different values for registers, inputs or outputs. A maximum of 10 different images can be displayed with one Multi Control.
We want to use this Multi Control now to do a progress bar, which then shows the progress of the Stopwatch.
But first thing we need are the different lengths of bars. We recommend you to do them on MS Paint starting with a 50x50px box going up to 500x50px by increments of 50px. Make sure to create a files readable for the controller.
Then input 3 Multi Controls to the Page and put labels above the Multi controls for the user to understand what the progress bar means.

## Multi Control

Set the DataType to Numeric Register and the DataIndex to 50, 51 and 52. (We will use them later in the program)
Then in the Multi Change, set the Data Nbrs 1-9.




Make sure to set these for every Multi Control.
For Data No. 1 should be used the 50xx50px bar, for No. 2 the 50x100px etc.
As for the ToggleLamp controls, change the monitor settings the same way as for the lamps.
And make sure that your Multi commands are actually big enough to fit the progress bars (50x500px)

Make sure you link also your buttons to the right DIs.
Now you only have to save and upload the page. Remember that you need to upload also the bars you created on Paint.
For now, this user frame will not work on with the current program. You can start, stop and reset the timer, but as there are no outputs on the registers we set for the Multi commands, they won't change.
Make a copy of the I_O_TIMER to I_O_TIMER_PROG_BAR. Try and change the program in a way, that it will output the seconds from 1 to 9 on R[50], every tenth of seconds on 51 (10-50) and every minute on 52.
Tip: You will need to change the REGISTER_TO_BINARY to a program that splits up the register into minutes, 10ths of seconds and seconds and then places them in the right register.
If you don't know how to do this program, go to the appendix where we have put a working program with explanations.
Now as you have created your program, call your page in the beginning of your program (normally it should be no. 7).
Try your program and user frame and change it if you like.

## Recapitulation

In this exercise you should have seen the basic Controls used to create customized webpages on the iPendant.
You should be able to create, save, upload and display your own webpages.
You should know how to call a specific webpage during a program.

## Appendix

### Main Program

I_O_TIMER_PROG_BAR 1/35

```
   1:  !Call user defined webpage
   2:  CALL DSP_WEBP(7)
   3:  !Reset the Timer
   4:  R[34:Timer]=0
   5:  LBL[2]
   6:  !DI101 starts timer
   7:  IF DI[101:Switch 1]=ON,JMP LBL[1]
   8:  !DI102 stops program
   9:  IF DI[102:Switch 2]=ON,JMP LBL[5]
  10:  !DI103 resets timer
  11:  IF DI[103:Switch 3]=ON,JMP LBL[4]
  12:  !Wait for not ovrloadng Processor
  13:  WAIT    .10(sec)
  14:  !Jump to beginning
  15:  JMP LBL[2]
  16:  !Start the Timer
  17:  LBL[1]
  18:  WAIT   1.00(sec)
  19:  !Timer Value increased
  20:  R[34:Timer]=R[34:Timer]+1
```

I_O_TIMER_PROG_BAR 35/35

```
  16:  !Start the Timer
  17:  LBL[1]
  18:  WAIT   1.00(sec)
  19:  !Timer Value increased
  20:  R[34:Timer]=R[34:Timer]+1
  21:  !Transform Reg to Prog Bar
  22:  CALL REG_TO_PROG_BAR
  23:  IF DI[102:Switch 2]=ON,JMP LBL[5]
  24:  IF DI[101:Switch 1]=OFF,JMP LBL[2]
  25:  JMP LBL[1]
  26:  !Reset Timer
  27:  LBL[4]
  28:  R[34:Timer]=0
  29:  R[50:1sec]=0
  30:  R[51:10sec]=0
  31:  R[52:1min]=0
  32:  JMP LBL[2]
  33:  !End of Program
  34:  LBL[5]
[End]
```

### Sub Program

REG_TO_PROG_BAR 1/28

```
   1:  !Set Registers
   2:  R[55:1minP]=0
   3:  R[54:10secP]=0
   4:  R[53:1secP]=R[34:Timer]
   5:  LBL[1]
   6:  !If Timer over 1 minute
   7:  IF R[53:1secP]>=60,JMP LBL[2]
   8:  JMP LBL[3]
   9:  !Minute Reg+1,Second Reg-60
  10:  LBL[2]
  11:  R[55:1minP]=R[55:1minP]+1
  12:  R[53:1secP]=R[53:1secP]-60
  13:  !Timer maybe still>1 minute
  14:  JMP LBL[1]
  15:  LBL[3]
  16:  !If Timer over 10 seconds
  17:  IF R[53:1secP]>=10,JMP LBL[4]
  18:  JMP LBL[5]
  19:  !10 sec Reg+1, Second Reg-10
  20:  LBL[4]
```

REG_TO_PROG_BAR 28/28

```
   9:  !Minute Reg+1,Second Reg-60
  10:  LBL[2]
  11:  R[55:1minP]=R[55:1minP]+1
  12:  R[53:1secP]=R[53:1secP]-60
  13:  !Timer maybe still>1 minute
  14:  JMP LBL[1]
  15:  LBL[3]
  16:  !If Timer over 10 seconds
  17:  IF R[53:1secP]>=10,JMP LBL[4]
  18:  JMP LBL[5]
  19:  !10 sec Reg+1, Second Reg-10
  20:  LBL[4]
  21:  R[54:10secP]=R[54:10secP]+1
  22:  R[53:1secP]=R[53:1secP]-10
  23:  JMP LBL[3]
  24:  !Set final Registers
  25:  R[50:1sec]=R[53:1secP]
  26:  R[51:10sec]=R[54:10secP]
  27:  R[52:1min]=R[55:1minP]
[End]
```

FANUC

Exercise 11

# Macro

FANUC

Exercise 11

# Macro

## Table of contents

## Equipment

For this exercise, you need:

Education Cell

## Example of a macro

The Education Cell already comes with a macro. Press the “SHIFT” and “TOOL 1” key. You do not have to reset the faults or press the deadman switch. You can see that the gripper will close or open.

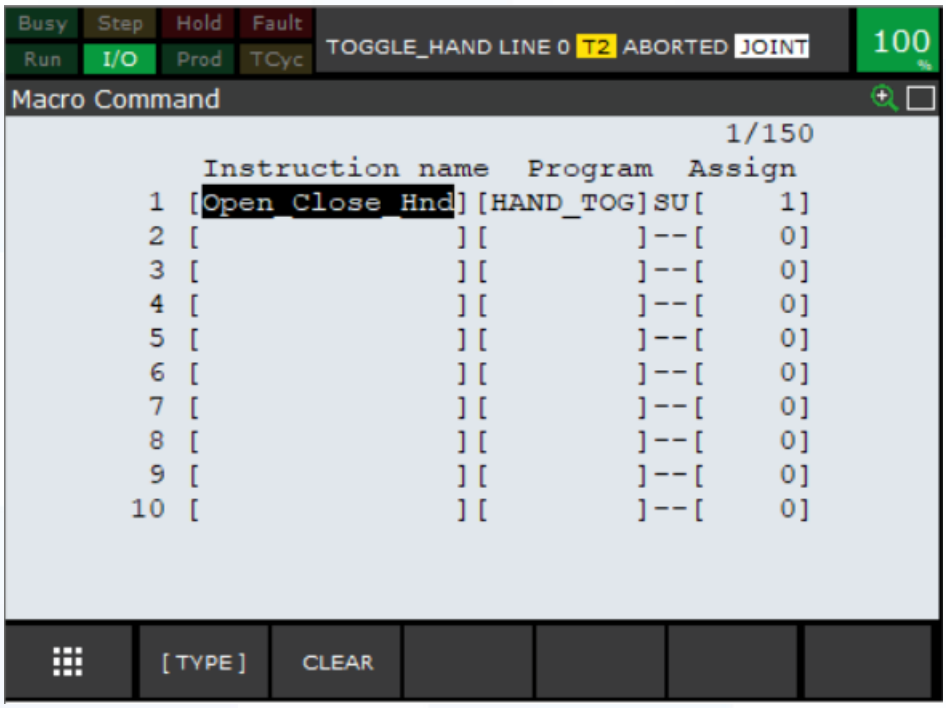

We will program the same program and use it for the “TOOL 2” key. Later you can program your own macro.

## Programming a Macro

A macro is programmed just like a normal program. Press the “SELECT” key and create a new program. Name the program “TOGGLE_HAND”. The program should open the gripper if it is closed and close the gripper if it is open.

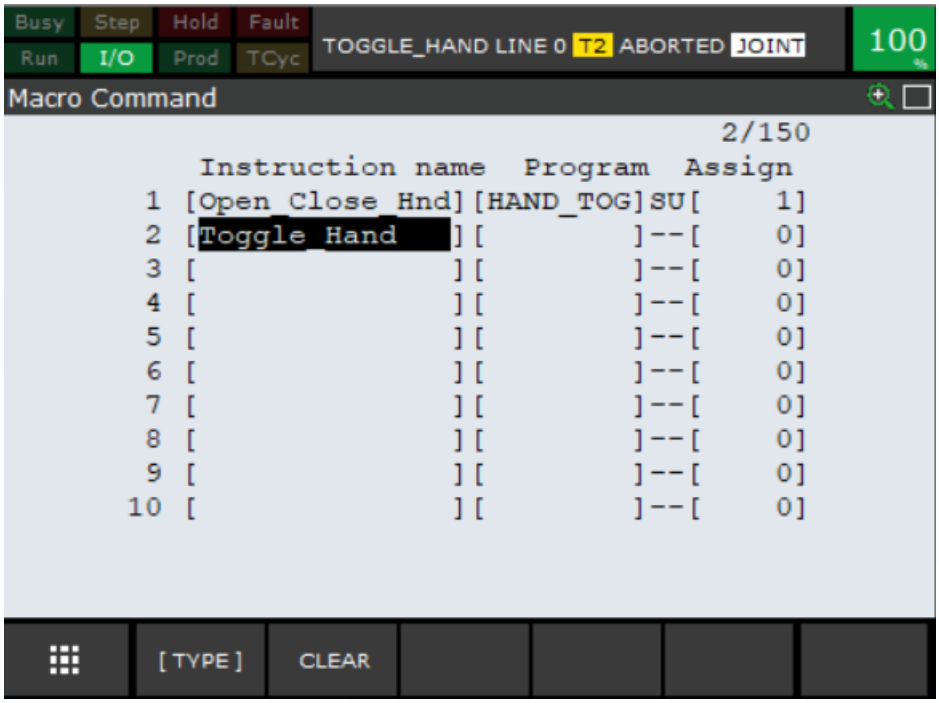
## Setting up a Macro

To set up a macro you have to go to the macro screen. Press on the Teach Pendant “MENU”, “6 SETUP” and then select “7 MACRO”.




On the first line we have the already set up macro. We will go to the second line and give the macro an instruction name, in this case we will call it Toggle_Hand (the instruction name does not need to be the same name as the program we will use).

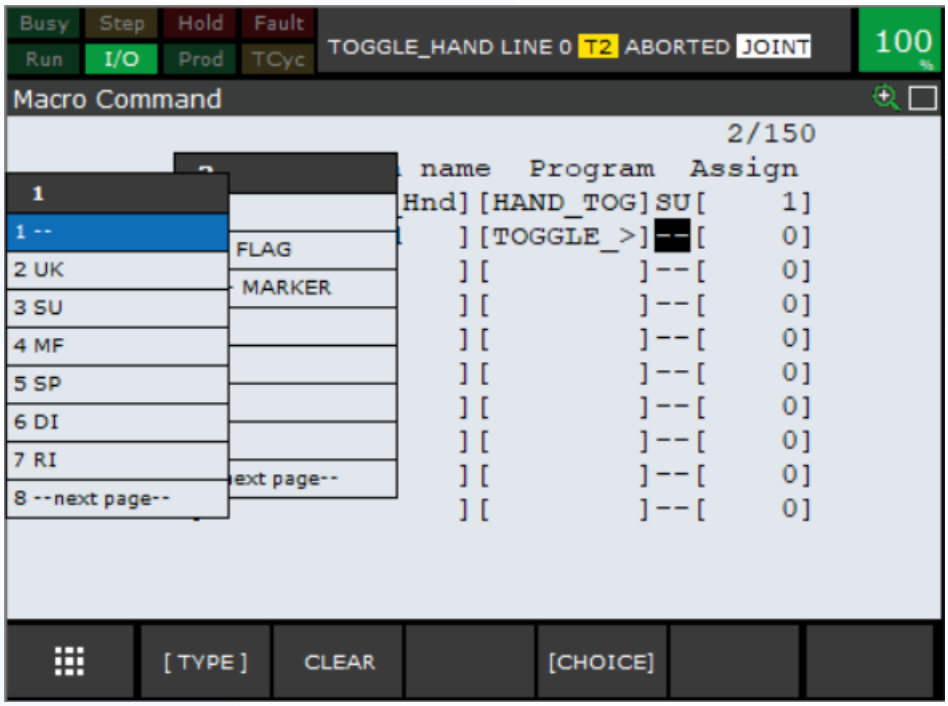
## Setting up a Macro



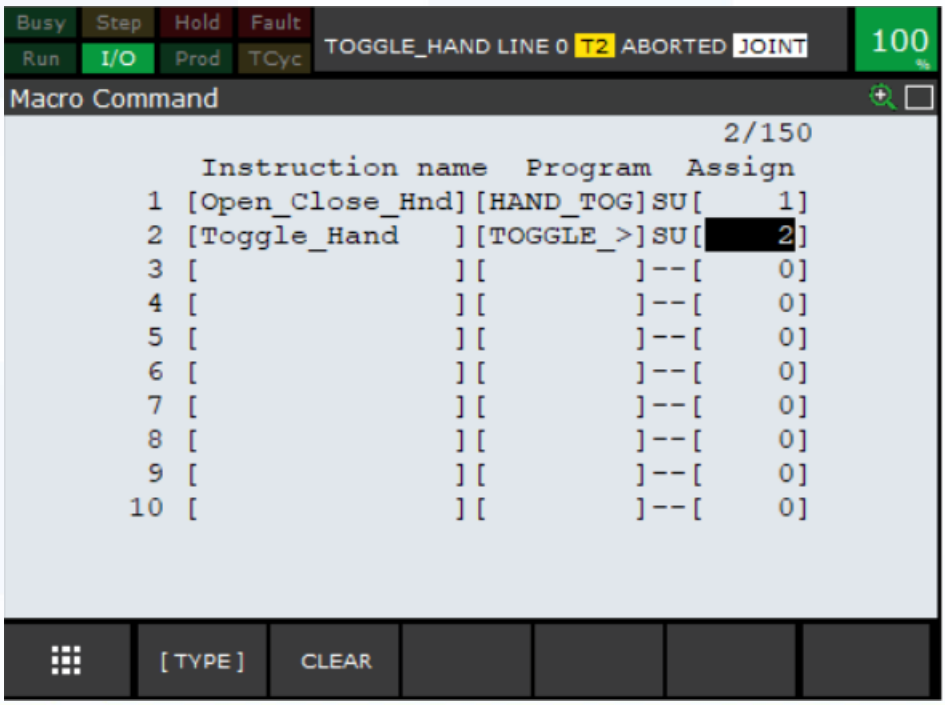

Now we have to select a program. Go under program and press F4 “Choice”, you can now select a program to use. The programs are alphabetically ordered, so search for our program “TOGGLE_HAND” and press “ENTER”.

## Setting up a Macro

Now we have to select the type of the macro, go to the '—' and press F4 "Choice"




Select 'SU' and for the assign number enter the number 2.



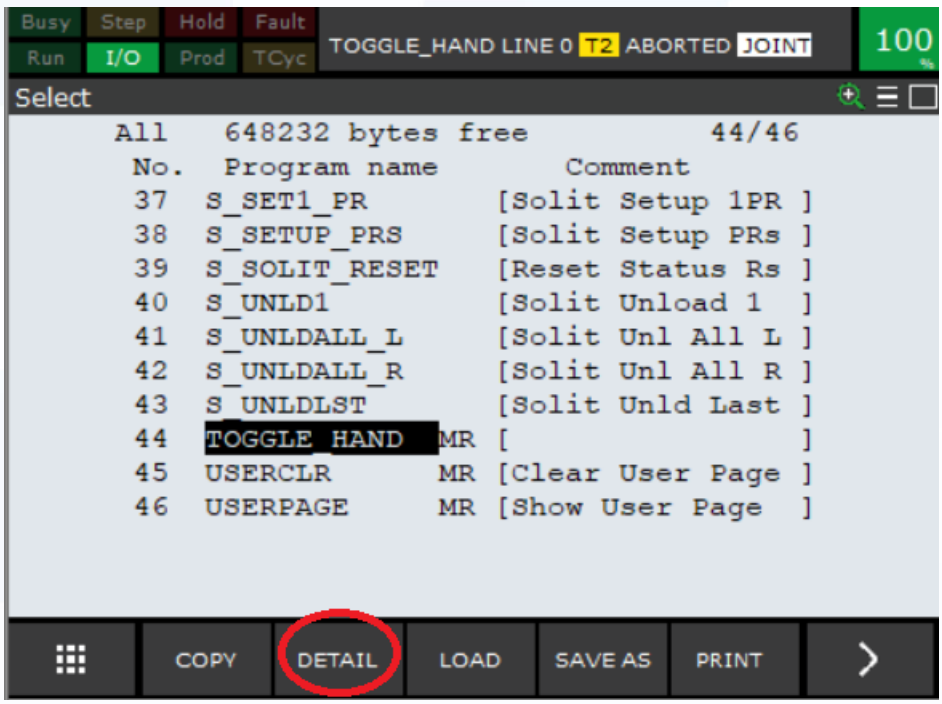
## Setting up a Macro

For more details refer to OM section 9.1.1 “Setting Macro Instructions”.

Now we have finished to set up the macro. Press the “SHIFT” and “TOOL 2” keys.

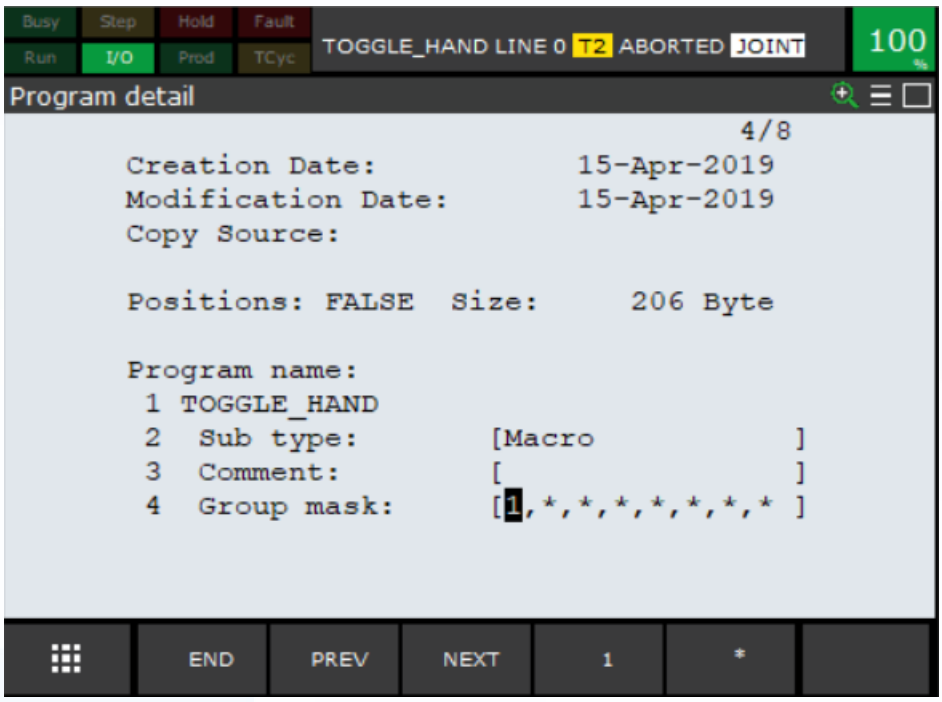
As you can see nothing is happening. We still have to press the deadman switch and then we can press “SHIFT” and “TOOL 2”.

To change that we have to change a setting in the program “TOGGLE_HAND”. The setting we are changing can only be changed if there is no motion instruction (you can use motion instruction in a macro but always set the UFrame and UTool in the program, also be careful on the speed of the motion).
Go to the program, press the “NEXT” key and press F2 “Detail”.

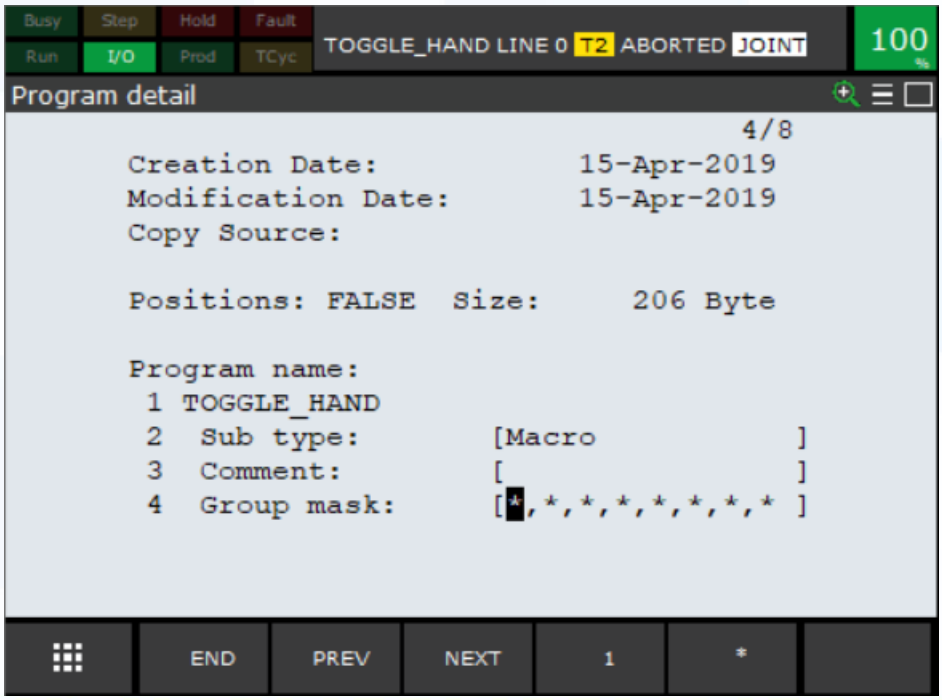


## Setting up a Macro

Scroll down to ‘Group Mask’.




Now we have to change the 1 to an asterisk ‘*’. Move the cursor to the 1 and press the F5 button to change it to ‘*’.



## Setting up a Macro

Press F1 “End” to go back to the select screen and now the macro should work as planned.
Press the “SHIFT” and “TOOL 2” key and see how the gripper closes and opens.
Only one Macro can be used on each “User” key.

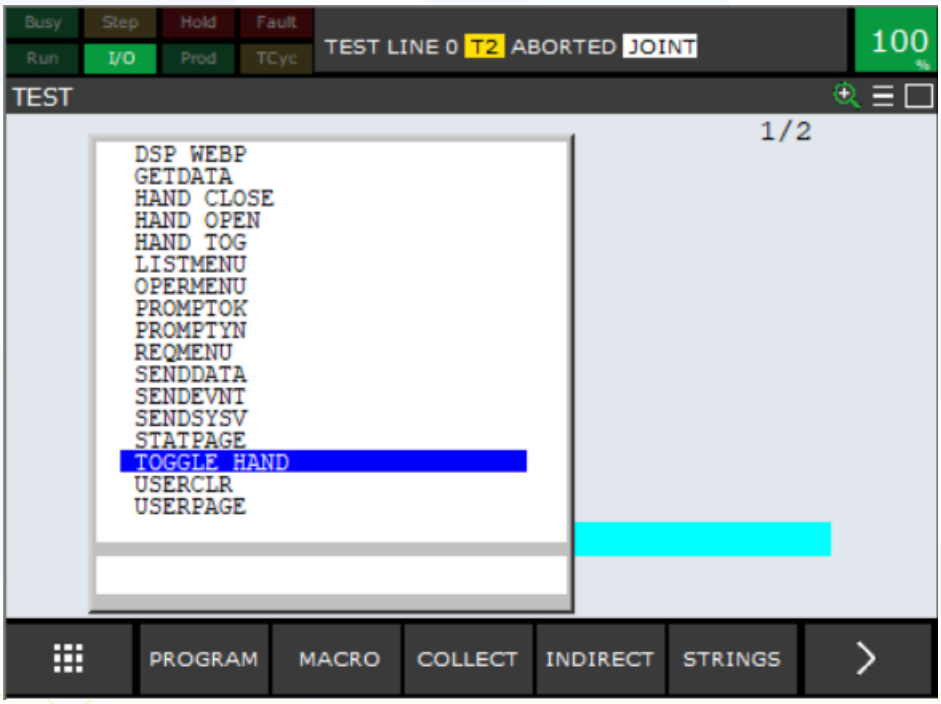
For more details on executing a macro refer to OM section 9.1.2 “Executing Macro Instructions”.

## Macro type MF

Instead of the macro type ‘SU’ you can also use ‘MF’. To use ‘MF’ macros you just need to press the “TOOL 1 “ and select the macro you want, then press “SHIFT” and F3 “Exec”. Up to 10 macros can be assigned to each “User” key. You can assign the numbers 1-10 for “TOOL 1” and the numbers 11-20 for “TOOL 2”.

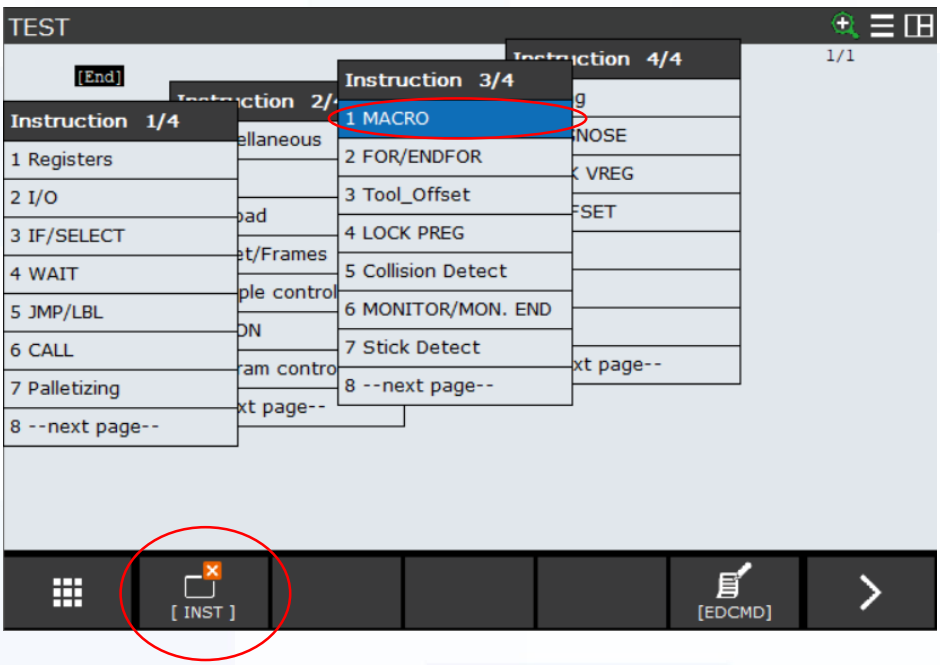
## Calling a Macro

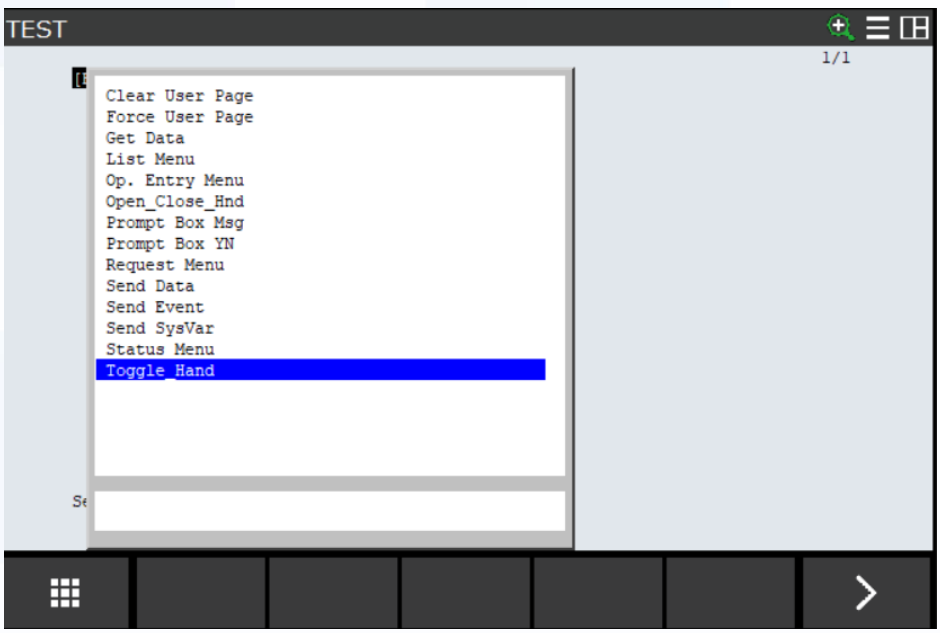
You call a macro in another program exactly like any other program. The only difference is that you have to press F2 for ‘macro’ in the program selection menu.

## Calling a Macro

Or if you have set up a macro in the macro screen then you can call it by pressing F1 ‘Inst’, go to the 3 page, select “MACRO” and chose the macro you need.



FANUC

Exercise 12

# Menu utility

FANUC

Exercise 12

# Menu utility

## Table of contents

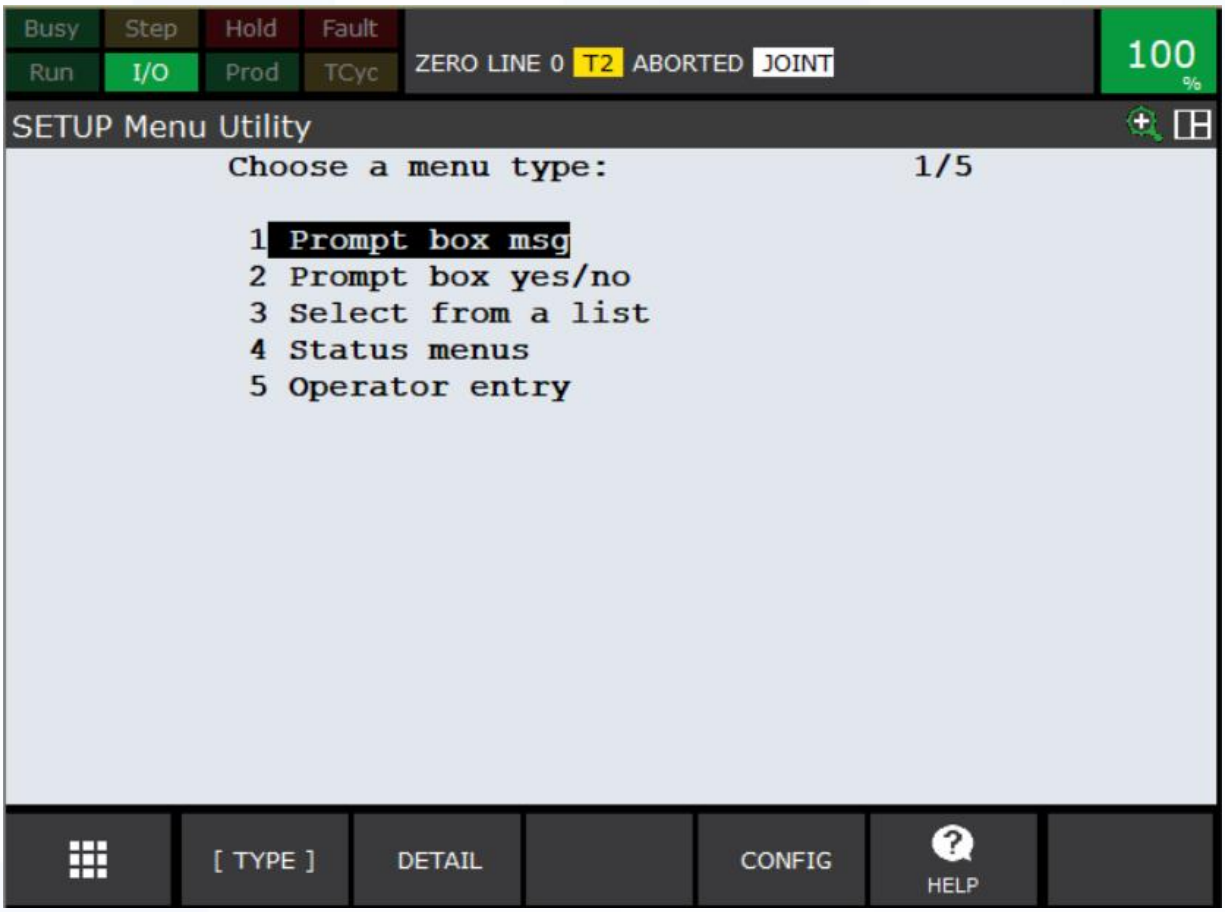
## Equipment

For this exercise, you need:

Education Cell

## Menu Utility Screen

To get to the Menu utility screen you have to press “MENU”, ‘6 Setup’ and then ‘3 Menu Utility’.
From there we can create and change some of these ‘utilities’.

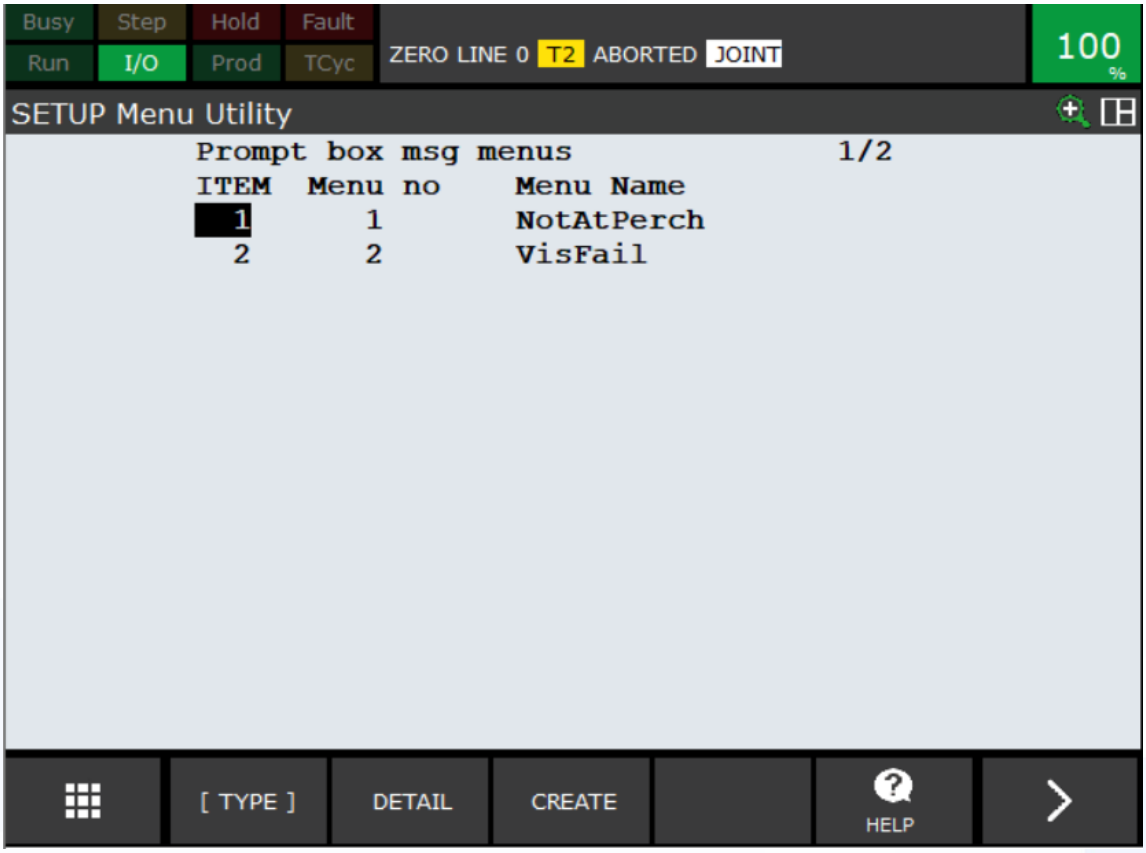


## PROMPTOK

PROMPTOK macro is used to display a simple pop up where the user has to press “ENTER” to proceed.

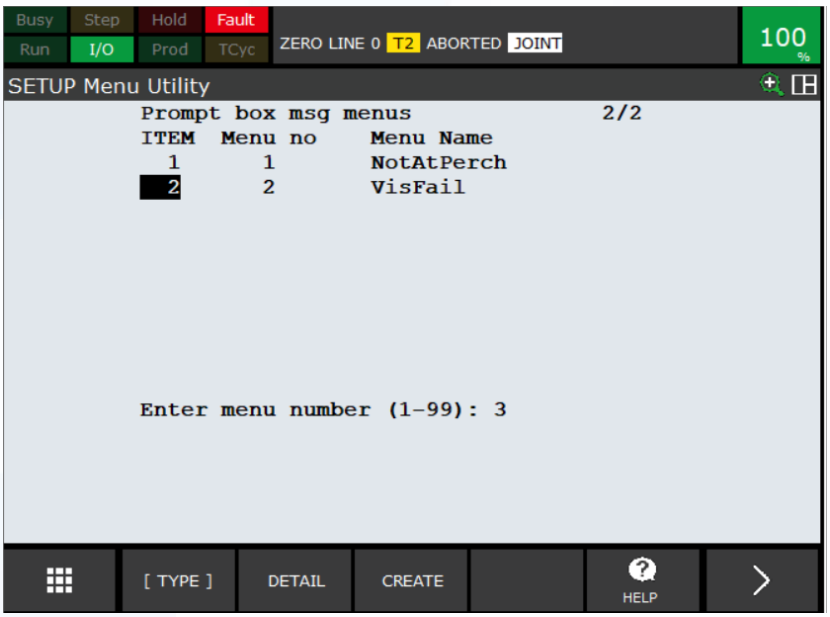
Press F2 ‘Details’ on Prompt box msg.

To create a new PROMPTOK press F3 ‘Create’. To make some changes on existing PROMPTOK press F2 ‘Detail’.

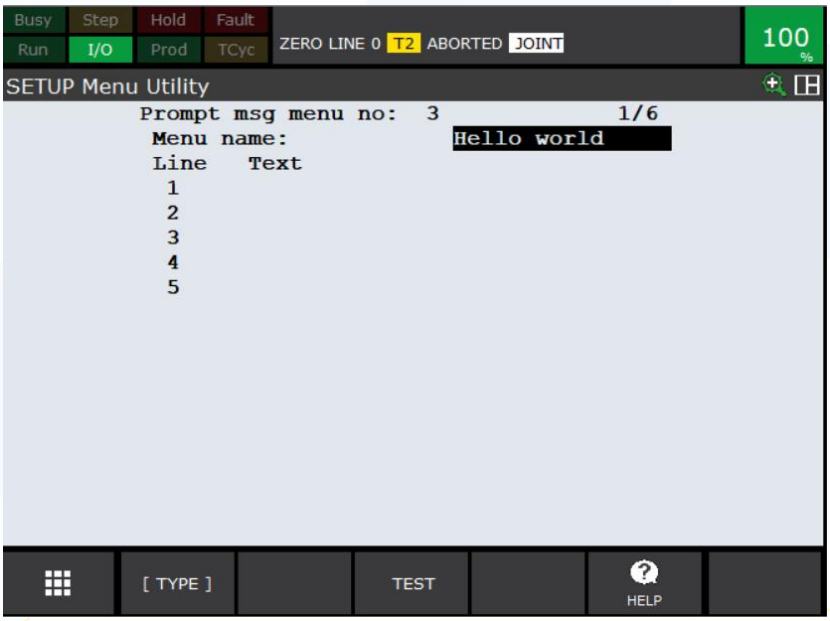


## PROMPTOK

To create a PROMPTOK you have to press F3 ‘Create’. You then will be asked which number it should have (every number can exist only once, it’s the ID number), we enter the number 3 (if not already taken).




We now have to give it a Menu name, in this example we will call it “Hello world”.



## PROMPTOK

In a PROMPTOK you can use 5 lines. Each line will only show the first 28 characters of the line.

As an example we put this sentence in the first line: “Hello world. How are you doing?”. As you can see there is the symbol “>” and this means that this is the maximum of characters which will be shown.
You do not need to fill every line.

For the second line enter “Fine” and for the third “And you?”.

## PROMPTOK

To test the PROMPTOK press F3 ‘Test’.


This screen will pop up and to proceed press the “ENTER” key.

Now you can change the first line to ‘Hello world. How are you?’, so the line is not cut off.

For further details see OM Optional Function 36.1.1 “Prompt Box Msg”.

To delete a PROMPTOK go back to the menu and go to the desired PROMPTOK and press “NEXT” and then F3 ‘Clear’.

## PROMPTOK

To call a PROMPTOK in a program, use the ‘CALL’ instruction. When you choose a program press F2 ‘Macro’ and select ‘PROMPTOK’.

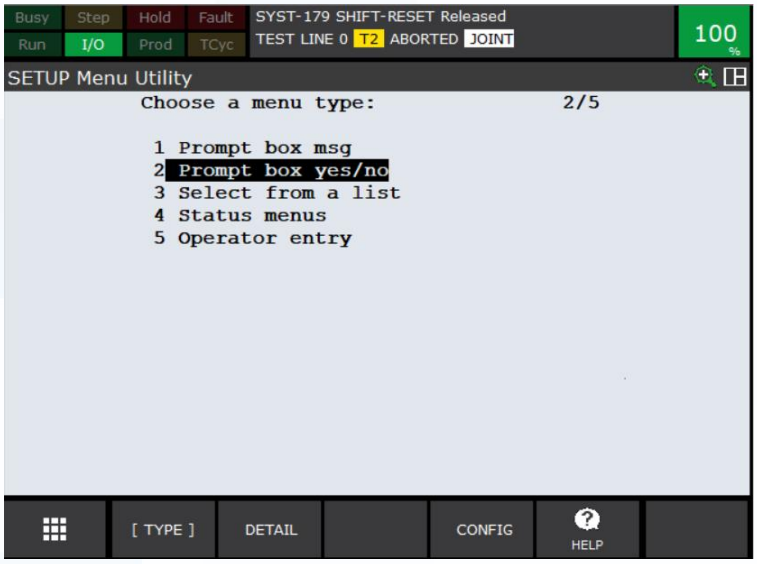

Now to define which PROMPTOK we want press F4 ‘Choice’, select ‘2 Constant’ and enter the number 3. The number we chose at the beginning of creating a PROMPTOK is the ID of the PROMPTOK and is needed to call the PROMPTOK.

Now you can start the program, the PROMPTOK will pop up on the Teach Pendant and the user has to press “ENTER” to continue.

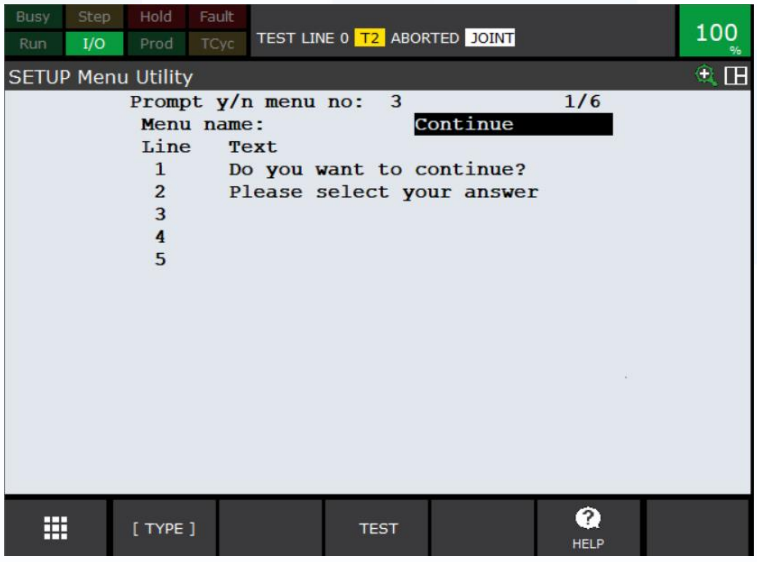
You can also call the PROMPTOK by going to the third page and selecting “MACRO” instead of “CALL” and select ‘Prompt Box Msg’.

## PROMPTYN

A PROMPTYN is a yes/no question which will be asked in a program.
Press F2 ‘Detail’ on ‘Prompt box yes/no’.

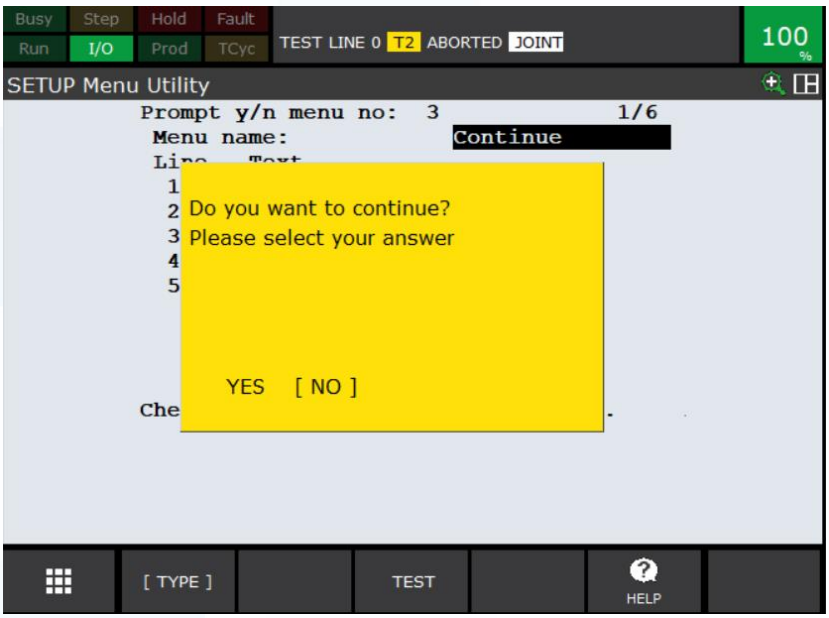



To create a PROMPTYN press F3 ‘Create’ and enter the number 3 (if not already taken).
We give it a Menu name and enter 2 lines. The lines work exactly like in PROMPTOK, they will only show 28 characters.



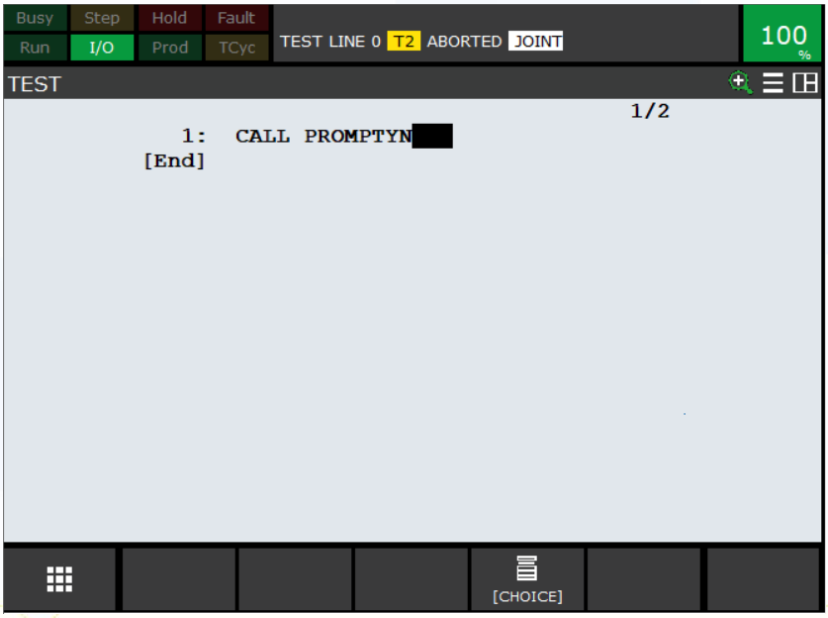
## PROMPTYN

Press F3 ‘Test’ to see the pop up. Use the arrow keys to navigate to ‘NO’ or ‘YES’ and press “ENTER” on your answer.




For further details see OM Optional Function 36.1.2 “Prompt Box YES/NO Menu”.

To call a PROMPTYN in a program use the “CALL” instruction and press F2 ‘Macro’ when you chose a program and select ‘PROMPTYN’.



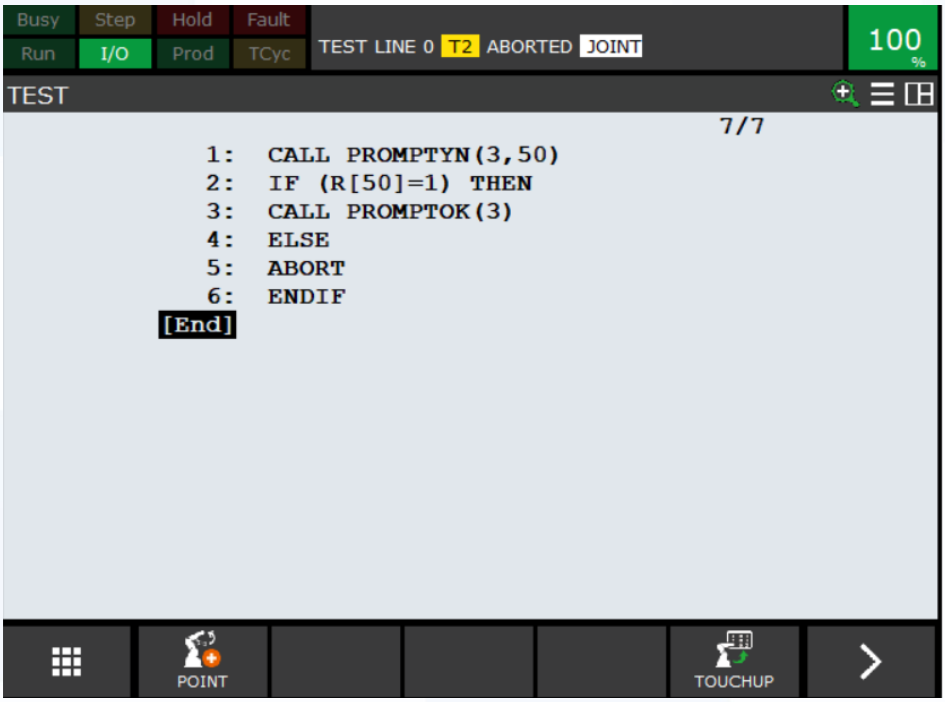
## PROMPTYN

Press F4 ‘Choice’, select ‘2 Constant’ and enter the ID number ,of our PROMPTYN, in this case 3.


Because PROMPTYN has an output we have to save this output on a Register, so press F4 ‘Choice’ again, select ‘2 Constant’ and enter the number 50 for R[50].
If the answer is ‘YES’ the value of R[50]=1 otherwise it will be 0.

## PROMPTYN

Now we program with the ‘IF’ function what will happen when we answer the PROMPTYN. If the answer is ‘YES’, we will call the PROMPTOK, we created previously, if ‘NO’ we abort the program. For the instruction ‘Abort’ press F1 ‘INST’, ‘8 next page’, ‘7 Program control’ and then select ‘Abort’.



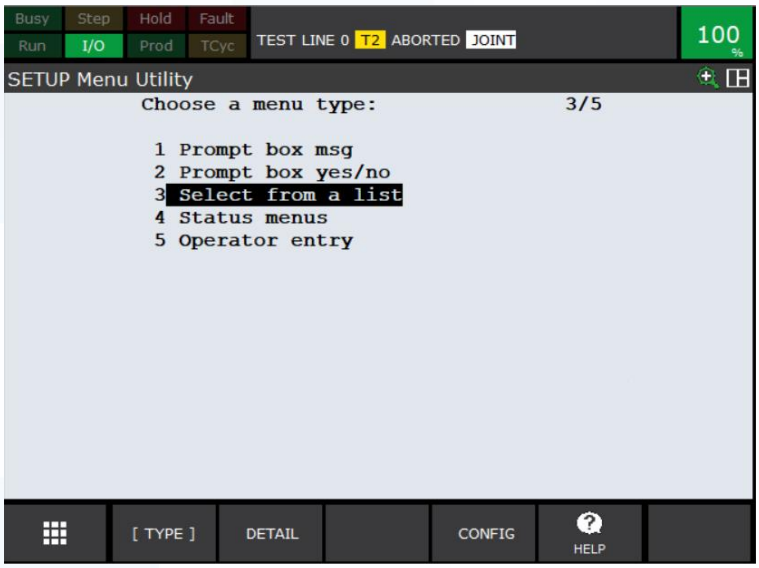

For further details on the ‘Abort instruction’ see OM section 4.14.2 “Abort Instruction”

Now you can test the program. Save this program because we will use it in the next example.

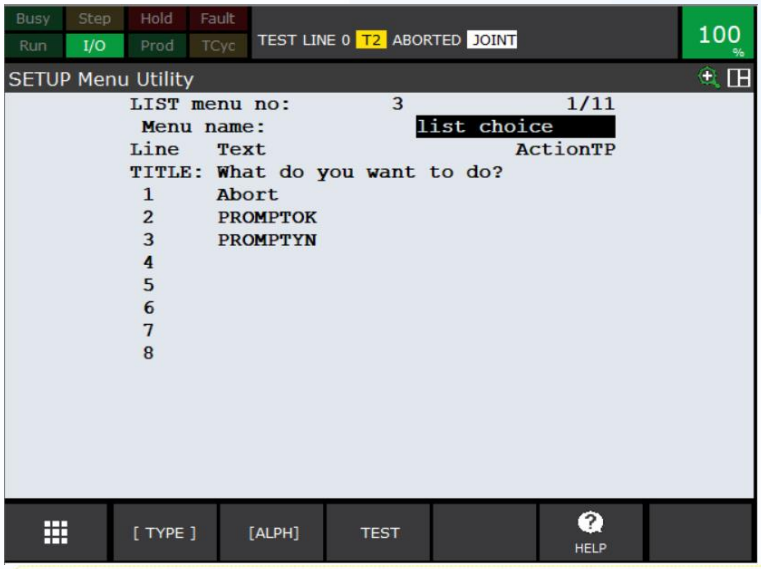
## LISTMENU

A LISTMENU gives you a list where you have to choose an answer between the number 1-8.

Go to the menu utility screen and press F2 ‘Detail’ on ‘Select from a list’.

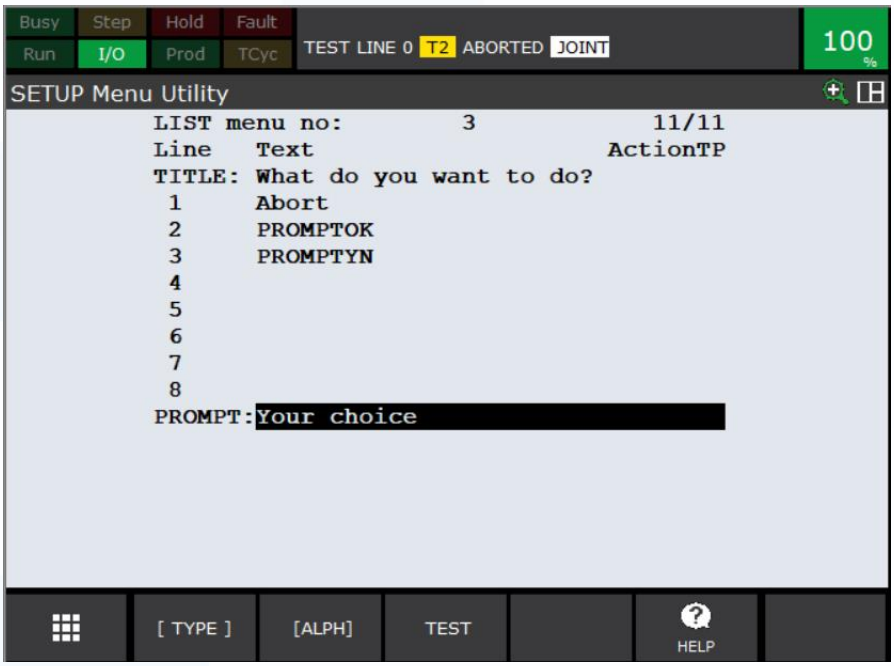



To create a LISTMENU press F3 ‘Create’ and give it the number 3 (if not already taken). We will give the LISTMENU the name ‘list choice’, the Title will be “What do you want to do?” and the first line will be ‘Abort’ second ‘PROMPTOK’ and third ‘PROMPTYN’. Each line will display 28 characters.

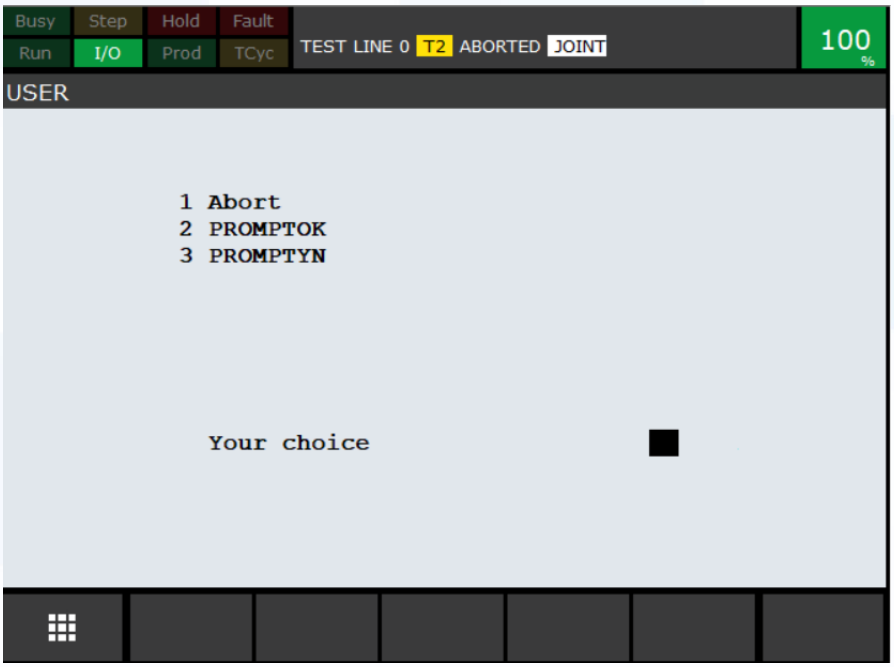


## LISTMENU

Now scroll down to ‘PROMPT’ and enter “Your choice”. This ‘PROMPT’ is only for what will be displayed where the user enters his choice.

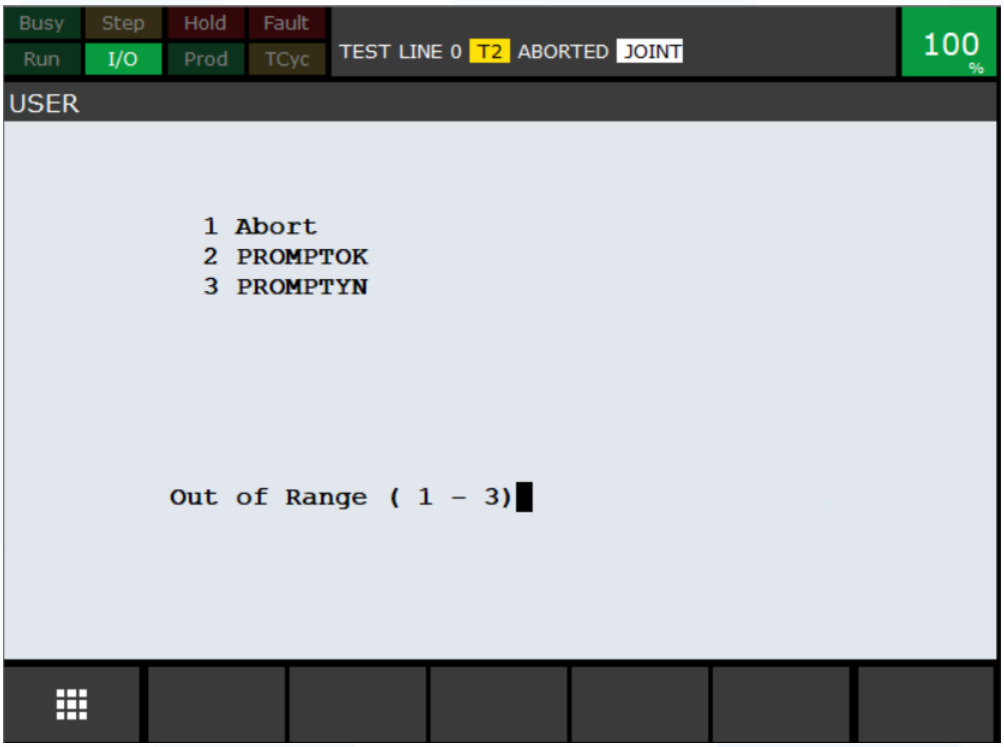



Now test the program with F3 ‘Test’ and enter your number of choice.



## LISTMENU

If you try to enter a number higher than 3 the program will say ‘Out of Range (1 – 3)’. This is to be sure the user does follow the list and the program can be executed as programmed. You can use up to 8 different lines to get an answer.
As you can also see the ‘Title’ will not be displayed but later when we call it in a program it will be displayed.




For further details see OM Optional Function 36.1.3 “List Menu”.

## LISTMENU

To call a LISTMENU use the “CALL” instruction. Just like previously press F2 ‘Macro’ when you are choosing a program and select ‘LISTMENU’.

Now press F4 ‘Choice’, select ‘2 Constant’ and enter the ID number 3. Then for the register which will save the answer, we will use R[51]. If you use a LISTMENU always be sure how many choices there are and what each choice should do. Press again F4 ‘Choice’, ‘2 Constant’ and enter the number 51.

## LISTMENU

To program it we can use the ‘IF’ condition but when we need to compare a lot of answers the ‘SELECT’ condition is faster (It compares every answer in one go and not one step after another like with the ‘IF’ condition.)

For further details on the conditional branch instruction ‘Select’ see OM section 4.7.4 “Conditional Branch Instruction”

To insert the ‘Select’ condition press F1 ‘INST’, ‘3 IF/SELECT’, ‘8 next page’ and then ‘Select R[ ]=…’. Enter the number 51 for R[ ] and as the result 1. For the first result the program should abort so create first another program with the abort instruction which we can call. Or use the JMP LBL[] instruction.

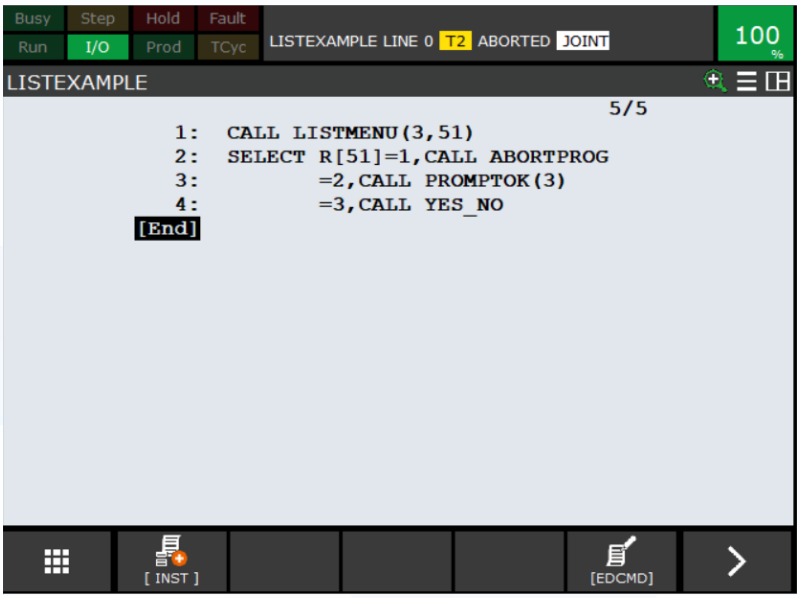
## LISTMENU

For the second result go to the next line press F1 ‘INST’, ‘3 IF/SELECT’. ‘8 next page’ and then ‘5 <select> =…’. Because we still use R[51] we do not need to clarify that again and can just enter the next result which is 2 in the ‘…’. The program we call does need an ‘Argument’, so select ‘3 CALL program()’ otherwise we cannot enter the number for PROMPTOK(3).

For further details on Arguments see OM section 4.7.6 “Arguments”


Now we do that for the third result but we will use the program we created with the PROMPTYN, so we do not need an argument.

## LISTMENU




The program is now finished and you can try it out.

Now you can try to program your own programs with PROMPTOK, PROMPTYN and LISTMENU.

# One common servo and control platform –
Infinite opportunities
**THAT's FANUC!**

WWW.FANUC.EU

**FA**
CNCs,
Servo Motors
and Lasers

**ROBOTS**
Industrial Robots,
Accessories
and Software

**ROBOCUT**
CNC Wire-Cut
Electric Discharge
Machines

**ROBODRILL**
Compact
CNC Machining
Centres

**ROBOSHOT**
Electric CNC
Injection Moulding
Machines

**ROBONANO**
Ultra Precision
Machine