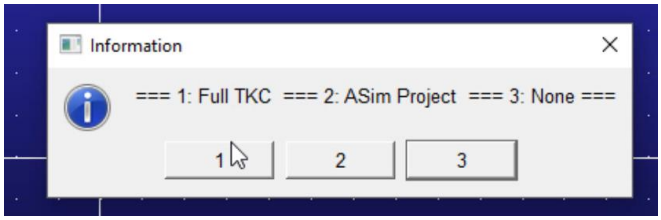
Starting TKC Session and Model File I/O

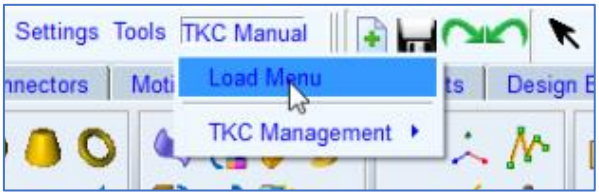
Table of Contents

1: Introduction and starting ADAMS

In this section we will illustrate the different options for starting TKC ADAMS, creating models and saving the required and the generated model data. TKC starts as a normal MSC/ADAMS/View session and loads the dedicated TKC functionality by selecting one of three options in the TKC loader dialog.



Thus, users can select to use ADAMS/View with *Full TKC*, a limited version called *ASim_Project* or *None* to not use TKC in the current ADAMS session. After the start-up phase of ADAMS/View, TKC is available selecting option one but has not been loaded yet unless the Auto Start option is activated. Manual loading of TKC is done by pressing the *Load Menu* button on the main *TKC Menu*.



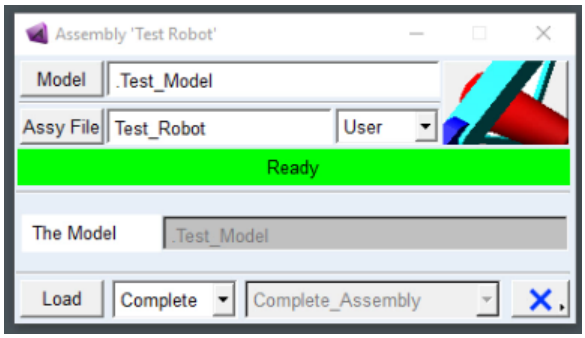
Next, the different toolboxes will be loaded into ADAMS/View either from a binary file or ASCII files. The latter option is slower but has to be performed only once to create a Toolkit binary file.

After loading all the required objects, the information dialog displays what is available now in our current ADAMS session. In case the dedicated *TKC Ribbon Bar* is loaded, two additional ribbons are created at the left side of the ribbon bar. One ribbon with groups of TKC system objects and one with a selection of toolkits loaded for this project.

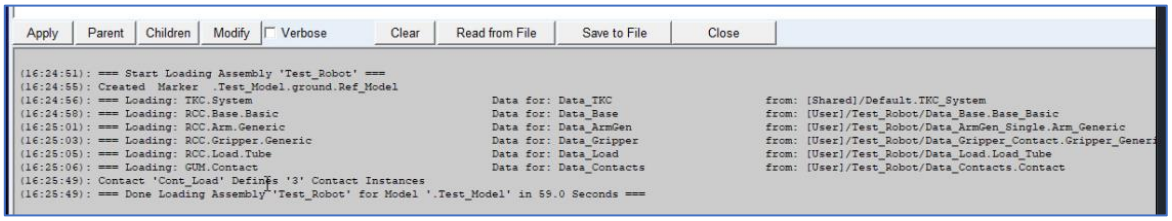
2: Open Assembly loader dialog and load model Assembly

Starting from here will load a model using the Assembly loader, the Assembly loader dialog is opened by clicking the third button from the left on the main Toolbar.

The Assembly loader is a generic dialog; it allows users to load objects from a range of different libraries. In the example shown, an assembly called *Test_Robot* from the User library.

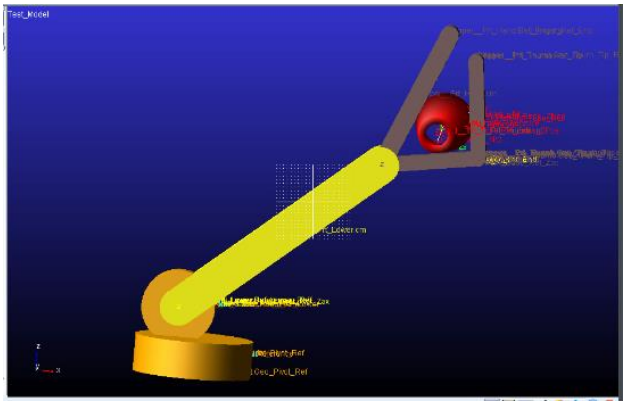


The different options and field of this Assembly loader dialog will be explained in next sections. For now, we will just load the Assembly in the current model by clicking on *Load*. The information panel echoes the different objects created in the loading process.



3: Close Assembly dialog and check model properties

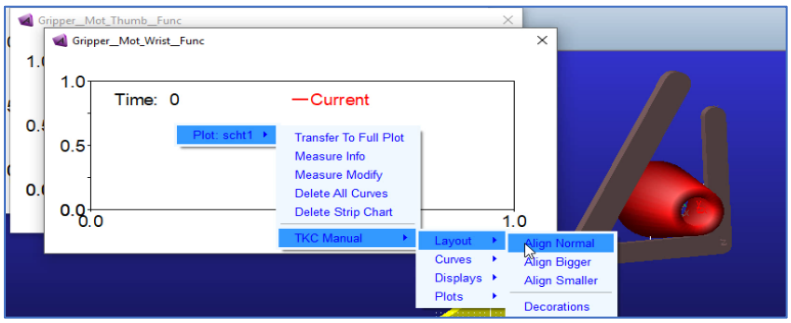
After the model objects are loaded into the model, we can check which data is defined, and we can visually check the different components in the model. You see here that indeed a robot has been defined, including the gripper and the load connected through the gripper. Next, we can perform a simulation, because all robot components required are already defined by the Assembly.



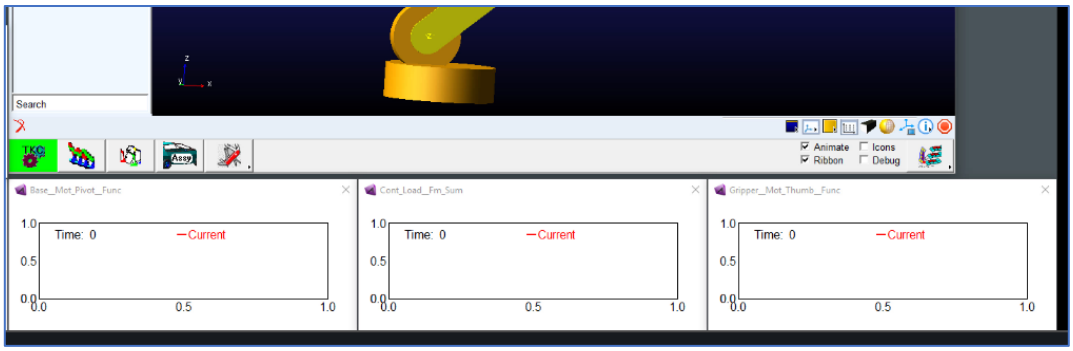
Once the Assembly is finished, we can close the dialog and start working on editing and adjusting the model to set it ready for performing a simulation. A smart thing to do is toggle the icons, so we do not have the icon names confusing us.

4: Perform a test simulation

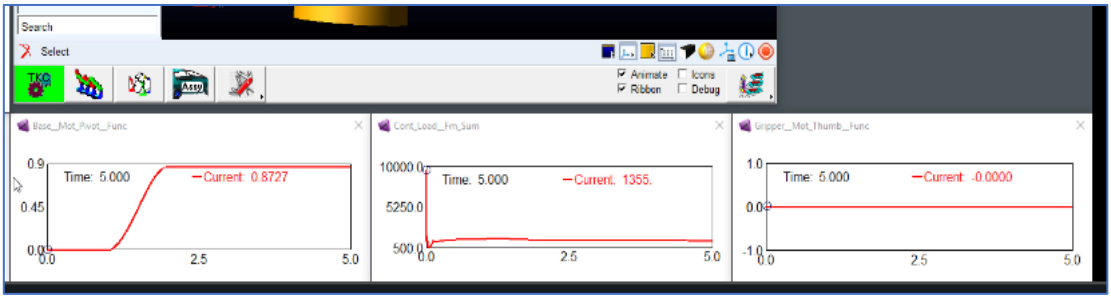
Next, we will display some signal measures defined in the model to track simulation signals.



Using TKC popup menus, measures can be aligned on the screen to obtain a clean screen layout.



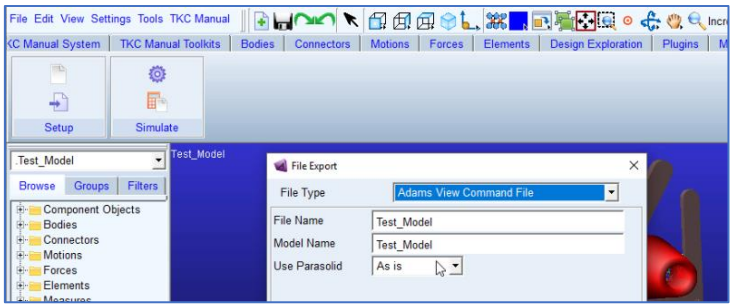
Now, we can open the Simulation Control dialog to perform a simulation. Set the End Time of the simulation to 5 secs and the number of Steps to 200 and perform a simulation.



The simulation results show the measure *Base__Mot_Pivot__Func* or the *Func*tion for the *Mot*ion angle of the *Pivot* of the robot *Base* part and *Cont_Load__Fm_Sum* which is the *F*orce *m*agnitude *Sum* of all *Cont*act forces between *Load* and gripper. Note the double underscore in the names of the measures that indicate a hierarchy level in a component.

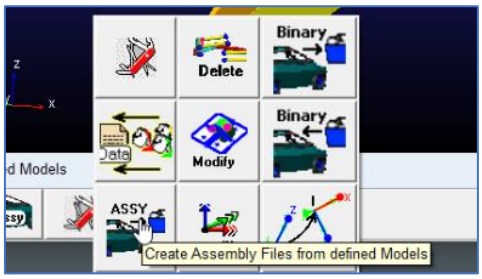
5: Save model in different file formats

Several methods are available to store TKC based models to a file. Standard ADAMS/View offers three methods, of which the file export and the binary file save will be discussed here. The *File-Export* method creates an ASCII based ADAMS/View command file. In this case, we will select the proposed default file name from the name of our model. This creates the file *Test_Model.cmd*.

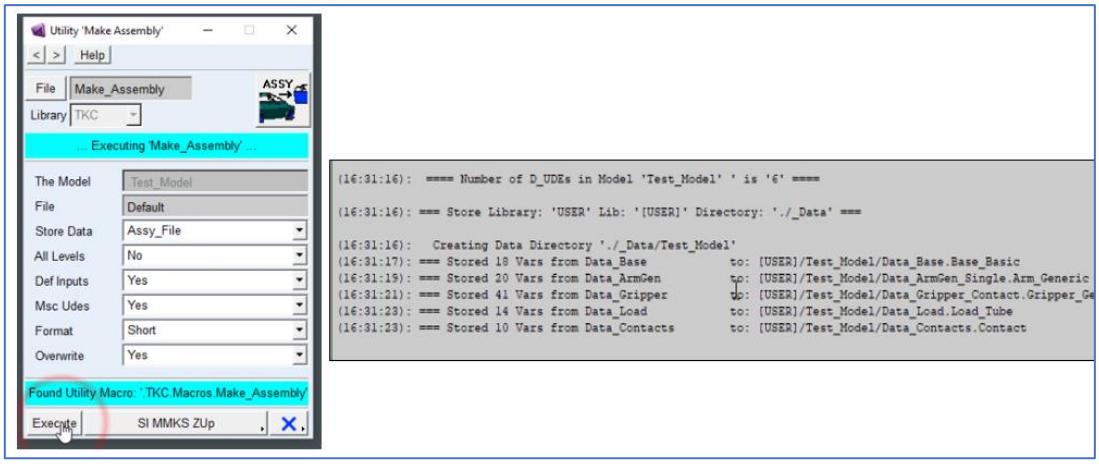


As a second option, the model can be stored in a binary file containing a full snapshot of the model state, the program GUI (Graphical User Interface) as well as the simulation results. The default name for this binary file, based on the model name is *Test_Model.bin*. As one may expect the size of the binary file is much bigger than that of the *.cmd file.

The third option to save our model is introduced by TKC, as it is an export of the model to a new TKC assembly. This is done by using the *Make_Assembly* Utility in the stack of Preferred Utilities in the Main Toolbar. This utility will extract all TKC model objects and create a new Assembly file from the model components defined in the model.



When using *File=Default*, the current model name is used as the base name of the assembly file *Test_Model.mac_assy*. By selecting Store_*Data=Assy_File,* model data files will be created in a data sub-directory named after the assembly (i.e., *Test_Model*). After executing the Utility Dialog, variable data files are created for all model data objects as well as a new Assembly file that can be used to recreate the model.



The image below shows a section of the assembly file. The data objects (Create_D_UDE), the backbone marker(s) and the group of component objects (Create_C_UDE) are visible.

Test_Model.mac_assy

```
!
TKC  Load_Utility  File      = "SI_MMKS_ZUp"
TKC  Load_Data     Data_Name = "$The_Model.Data_TKC"  File = "Default"
!
TKC  Create_D_UDE  Name = "Data_Base"        Type = "RCC.Base.Basic"          File = "Test_Model/Data_Base"
TKC  Create_D_UDE  Name = "Data_ArmGen"      Type = "RCC.Arm.Generic"         File = "Test_Model/Data_ArmGen_Single"
TKC  Create_D_UDE  Name = "Data_Gripper"     Type = "RCC.Gripper.Generic"     File = "Test_Model/Data_Gripper_Contact"
TKC  Create_D_UDE  Name = "Data_Load"        Type = "RCC.Load.Tube"           File = "Test_Model/Data_Load"
TKC  Create_D_UDE  Name = "Data_Contacts"    Type = "GUM.Contact"             File = "Test_Model/Data_Contacts"
!
!
! ================================================================
! ===  Block  Back_Bone ! Define the Registered Markers ===========
! ================================================================
!
TKC  Set_Marker  Marker = "$The_Model.ground.Ref_Model" &
     Loc  = "({0.0, 0.0, 0.0}m)" &
     Ori  = "({0.0, 0.0, 0.0}d)"
!
!
! ================================================================
! === @Block  Components ! Define the Component UDE  Objects ======
! ================================================================
!
TKC  Create_C_UDE  Name      =  "Base"            &
                   Type      =  "RCC.Base"        &
                   Data_Name =  "Data_Base"       &
                   Inputs    =  &
                  "Reference  =  $The_Model.ground.Ref_Model", &
                  "Carrier    =  $The_Model.ground"
!
```

6: Reload the model from the created assembly

At this point, the robot model can be re-created from the assembly file *Test_Model.mac_assy*. To test this, delete the current model in the ADAMS session and create an empty new model *Model_2*. By loading the *Test_Model* Assembly in the same way as the original *Test_Robot,* we confirm that the *Test_Model.mac_assy* is indeed a functional TKC Assembly. Note that TKC components added on-the-fly to model *Test_Model* after loading Assembly *Test_Robot* would also have been saved to Assembly *Test_Model*.

Loading the model data from the ADAMS/View command file is a standard ADAMS operation executed using the *File Import* dialog.

In this section, we have illustrated the different ways to load models into TKC and write to file. Summarising, TKC users can still use the standard ADAMS/View functionality of saving a command file or binary file and loading it back into ADAMS.

A final message is that models stored as TKC command files can be run without the TKC functionality, as it is a plain ADAMS/View command file. The TKC functionality layer is stored in the model as properties of design variables which only become active inside the TKC environment. Without TKC, the model can be modified and simulated as any other standard ADAMS/View model. The TKC layer introduces a more functional and user-friendly interface to modify and manage model parameter values.