

Die neue Version

EASY-ROB™ V7.6



Dezember 2018

Version 1.6

EASY-ROB™

Inhaltsverzeichnis

EASY-ROB™ V7.6 Update.....	5
Ändern von 3D Geometrien zur Simulationslaufzeit.....	6
Neue API-Methode "body_renewing()"	6
Geometrie-Import: Optimierung in drei Schritten	7
Programmier-Beispiel zu "body_renewing()"	7
Neuheiten für EASY-ROB™ Robotics Simulation Kernel (ERK).....	8
ToolPath - Trajektorien leicht erzeugen	8
ToolBox - ein Leistungsstarke Ergänzung	10
Schematische Darstellung: ERK Workflow mit Zusatz APIs und Optionen	12
API-Post-Process - Roboterprogramme erzeugen	13
AutoPath™ als eigenständiges Modul verfügbar.....	15
Callback-Funktion	15
Direkter Bezug zur eigenen Anwendung	16
Kommunikation: Host-Applikation - ERK mit AutoPath™	16
Vorteile und Anwendungsmöglichkeiten	17
Feature-Übersicht	17
Privaten Benutzerschlüssel (OwnerKey) festlegen.....	18
Encrypten- Verschlüsselung durchführen	18
Decrypten- Verschlüsselung rückgängig machen	18
Wo ist der usr_ownerkey zu finden?	18
Benutzerschlüssel nur in EASY-ROB™ direkt veränderbar	19
Benutzerschlüssel ändern oder neu erzeugen	19
Verschlüsselung <i>ohne</i> Benutzerschlüssel	19
Verschlüsselung <i>mit</i> Benutzerschlüssel.....	19
Beispiel einer Verschlüsselung	20
Sicherheitshinweis	20
3D-PDF Export mit Animation als SDK.....	21
Programmierschnittstelle.....	21
Selbst-Kollision (Itself Collision).....	22
Selbst-Kollision global aktivieren	22
Kollisionsausschlusslisten für Roboter erzeugen.....	23
Individuelle Einstellungen über das 3D-CAD Window	25
Verbesserte Referenz-Kollision.....	26
„Referenz-Kollision“ aktivieren, Geometrien ausschließen	26
Parent- und Child devices	27
Vollständige Roboterbibliotheken	28
KUKA die neue CYBERTECH-Serie.....	29
Stäubli die neue TX-2 Serie mit CS9 Steuerung.....	30
Universal Robots die neue e Serie	31
Kawasaki die CP, CX und BX Serie.....	32
FANUC die R-2000 Serie.....	33
FANUC die Arc-Mate Serie	34
Kontakt.....	35
Eigene Notizen	36

EASY-ROB™ V7.6 Update

Hallo liebe EASY-ROB™ Community!

Wir freuen uns, die neue EASY-ROB™ Version 7.6 vorstellen zu dürfen und wie immer findet Ihr die Highlights hier in der Schnellübersicht:

- **Roboter Bibliothek- Top-Thema**

Die EASY-ROB™ Roboterbibliothek wächst kontinuierlich –

Zu Version 7.6 werden mehr als 20 neue Modelle veröffentlicht werden. Dabei wurden von unseren Mitarbeitern nicht nur einzelne Modelle, sondern auch ganze Serien integriert z.B. von KUKA die Cybertech Serie, von Stäubli die TX-2 Serie mit CS9 Steuerung und von Universal Robots die e Serie. Natürlich wurden auch viele Modelle von weiteren Herstellern integriert.

- **Ändern von 3D Geometrien zur Simulationslaufzeit**

Importierte Geometrien können über die API des Robotics Framework (DLL) und Robotics Simulator Professional (Exe) zur Simulationslaufzeit geändert werden. Damit lassen sich noch prozessnähere Simulationen erstellen.

- **EASY-ROB™ AutoPath™**

Kollisionsfreie Bahnplanung

EASY-ROB™ AutoPath™ erzeugt kollisionsfreie Bahnen. Die automatische Berechnung erleichtert die Arbeit des Bedieners erheblich. Lassen Sie auch Ihre Software von der neuen Programmierart profitieren.

- **EASY-ROB™ Selbst-Kollision**

Die Selbst-Kollision (Itself Collision) erlaubt es die Kollision des Roboters mit sich selbst zu prüfen, was sonst nur mit den Einstellungen über die Verfahrbereiche abgesichert wurde.

- **Sichere Dateien mit der EASY-ROB™ Verschlüsselung und Benutzerschlüssel**

Von nun an können Sie Ihre erstellten *.cel-, *.ras oder *.rob-Dateien vor fremdem Augen schützen. Entweder Sie verwenden die EASY-ROB™ Verschlüsselung oder kombinieren diese sogar mit Ihrem eigenen privaten „top secret“-Benutzerschlüssel.

Eins ist sicher: damit bleiben Ihre Daten sicher!

- **ERK Neuheiten: ToolPath, ToolBox und API-Post-Process**

Der EASY-ROB™ Robotics Simulation Kernel wurde massiv erweitert. So gibt es ein neues Grundmodul AutoPath, das dem allg. AutoPath in nichts nachsteht. Mit ToolPath wird eine weitere Methode zur Verarbeitung von Roboterzielpositionen angeboten. Weiter gibt es die neue leistungsstarke Option ToolBox und eine neue Schnittstelle API-Post-Process.

Ab sofort steht allen Kunden mit einer gültigen v7.6 Lizenz oder einem Softwarepflegevertrag die neue EASY-ROB™ Version 7.6 kostenfrei zur Verfügung.

Für Kunden älterer Versionen besteht die Möglichkeit ein Update zu erwerben. Nehmen Sie dazu bitte mit unserem Vertrieb unter +49 6192 921 70 79 oder sales@easy-rob.com Kontakt auf.

Für Ihre Anregungen und Verbesserungsvorschläge bedanken wir uns schon jetzt bei Ihnen.

Vielen Dank

Ihr EASY-ROB Team

Ändern von 3D Geometrien zur Simulationslaufzeit

Bisher konnten 3D Geometrien (z.B. STL und IGP Dateien) nur beim Import manipuliert werden. Danach war es nicht mehr möglich diese Geometrien in der Form (z.B. Skalierung) zu ändern.

Mit einer neuen API-Methode lassen sich Geometrien nun zur Simulationslaufzeit verändern, in dem die Punkte/Vertices "verschoben" werden. Das kann z.B. sinnvoll sein um einen Biegevorgang darzustellen, so dass der Roboter in jedem Prozessschritt an einem sich verändernden Bauteil mitfährt, oder um eine Geometrie schlichtweg nur zu skalieren.

Für jede Geometrie ist ein Speichermodell hinterlegt:

- [CAD_MEM_VBO](#)
Geometrie wurde optimiert und in die Grafikkarte geladen. (Vertex Buffer Objects - VBO)
- [CAD_MEM_GRAPHICS](#)
Geometrie wurde optimiert und ist intern durch Vertex-Arrays realisiert, keine VBO
- [CAD_MEM_NATIVE_CPU](#)
Geometrie nicht optimiert, verbleibt im ursprünglichen "nativen" Zustand.

Neue API-Methode "body_renewing()"

Die Methode "*ER_CAPI_CAD_IO::body_renewing()*" erlaubt es Geometrien zu erneuern und sie in die drei zuvor genannten Speichermodelle zu überführen. Hierbei kann auch entschieden werden ob die Geometrie in die Collision-List aufgenommen werden soll, das ist je nach Performance und Anwendung abhängig. Das geschieht mittels des Parameters "*renew_param*".

◆ body_renewing()

```
static ER_DllExport int ER_CAPI_CAD_IO::body_renewing ( void * body_handle,
                                                    int    cad_mem_model,
                                                    int    renew_param
                                                    )
```

static

Renews a body after modifying its vertices xyz values via API
Parameter *body_handle* must be valid and not 0
Param *cad_mem_model* determine the cad memory model and is one of [ER_CAD_MEM_VBO](#), [ER_CAD_MEM_GRAPHICS](#), [ER_CAD_MEM_NATIVE_CPU](#)
Param *renew_param* allows to control the renewing procedure
It is a bitwise inclusive OR operator (|) of [ER_CAD_RENEW_GEOMETRY](#), [ER_CAD_RENEW_COLLISION](#), [ER_CAD_RENEW_DEL_COLLISION](#)

Remarks
Setting the body to cad memory model = [ER_CAD_MEM_NATIVE_CPU](#) allows to modify its vertices and visualize it immediatly in the 3D Scene.

Parameters

[in] **body_handle** - handle to body
[in] **cad_mem_model** - bodies memory model
[in] **renew_param** - parameter to control renewing procedure

Return values
0 - Ok
1 - Error, renewing failed

Im Speichermodell CAD_MEM_NATIVE_CPU können die Vertices nun "direkt" via API geändert werden, **ohne** dass ein "body_renewing()" erneut aufgerufen wird. In diesem Fall sollen keine Kollisionen etc. geprüft werden.

Diese Neuheit steht nun für das Robotics Framework (DLL) und den Robotics Simulator Professional (Exe) zur Verfügung.

Geometrie-Import: Optimierung in drei Schritten

1. Schritt

Beim Importieren von Geometrien werden diese im ersten Schritt 1:1 abgelegt und können direkt, d.h. ohne Skalierung visualisiert werden. Hierbei werden auch Bounding-Boxes berechnet und die Geometrien befinden sich im Speichermodell **CAD_MEM_NATIVE_CPU**.

2. Schritt

Im zweiten Schritt werden die Geometrien bzw. dessen Objekte optimiert. Hier werden Vertices z.B. reduziert und für die OpenGL Rendering Arrays erzeugt.
Die Geometrien liegen nun im Speichermodell **CAD_MEM_GRAFICS** vor.

3. Schritt

Im letzten Schritt werden diese Arrays in die Grafikkarte geladen (Vertex Buffer Objects - VBO), falls möglich. Hier wirkt Speichermodell **CAD_MEM_VBO**.

Programmier-Beispiel zu "body_renewing()"

Im Folgenden einfachen Beispiel werden für eine aktuelle Geometrie die z-Werte aller Vertices verdoppelt bzw. im nächsten Aufruf wieder halbiert. Parameter "renew_param" wird = ER_CAD_RENEW_GEOMETRY | ER_CAD_RENEW_COLLISION gesetzt, so dass auch Kollision für die geänderte Geometrie getestet werden kann. Der grafische Update visualisiert das Ergebnis.

```
// Example:
void *body_hnd = er_cad_io.inq_body_handle_current(); // get body handle of current group
int n_obj = er_cad_io.inq_body_n_obj(body_hnd); // get number of objects within body
er_user_io_dialog_info_line_msg(0, "cBody With n_obj = %d", n_obj);
int z_sign = -1;
z_sign = -z_sign;
for (int i_obj=0; i_obj<n_obj; ++i_obj) // go for all objects
{
    void *obj_hnd = er_cad_io.inq_body_obj_handle(body_hnd, i_obj); // get object handle
    int n_pnt = *er_cad_io.inq_body_obj_n_pnt(obj_hnd); // get number of vertices of object
    for (int i_pnt=0; i_pnt<n_pnt; ++i_pnt) // go for all vertices
    {
        float *p = er_cad_io.inq_body_obj_points(obj_hnd, i_pnt);
        p[2] *= z_sign>0 ? 2.0f : 0.5f; // change z-value, just double or halve
    }
}

// Renew to see the effect, refresh complete geometry and the collision model as well
int cad_mem_model = er_cad_io.inq_body_cad_mem_model(body_hnd); // get current cad memory model
cad_mem_model = ER_CAD_MEM_NATIVE_CPU; // change cad memory model to ER_CAD_MEM_NATIVE_CPU

int renew_param = ER_CAD_RENEW_GEOMETRY | ER_CAD_RENEW_COLLISION;

int res = er_cad_io.body_renewing(body_hnd, cad_mem_model, renew_param);
if (res)
    _info_line_msg(0, "ERROR: body_renewing() failed");

// Visualize here
er_sys_view.grf_update_export();
...
```

Neuheiten für EASY-ROB™ Robotics Simulation Kernel (ERK)

In diesem Release wurde besonderes Augenmerk auf den Robotics Simulation Kernel (ERK) und eine vereinfachte Einbindung in technologiebasierte Software Applikationen gelegt.

ToolPath - Trajektorien leicht erzeugen

Der ToolPath ist eine Trajektorie für Roboter oder Geräte und besteht aus Zielpositionen (Targets) und kann im ERK erzeugt werden (Methodenklasse: ERK_CAPI_TOOLPATH). Alle Targets beinhalten Attribute wie Name, ID, Index, Geschwindigkeit, Beschleunigung, externe Achswerte, Tooldaten und -namen, Bezugskoordinatensysteme, Synchronisations-Flags, Bewegungsarten (PTP, SLEW, LIN, CIRC), Instructions, etc.

Die Anzahl von ToolPaths sowie die zugehörigen Targets ist unbegrenzt. ToolPaths sind dem Roboter eindeutig zugeordnet, können erzeugt, gelöscht und in der Reihenfolge verschoben werden.

Zielpositionen, lassen sich mittels umfangreicher Methoden vollständig manipulieren, d.h. man kann diese löschen, verschieben, ignorieren, neue erstellen, Änderungen in der Reihenfolge vornehmen und auf sämtliche Daten zugreifen.

Die implizite Verwendung von ToolPath-Templates erleichtert das Hinzufügen von Targets erheblich, so dass nur Daten wie beispielsweise Position und Geschwindigkeit vorgegeben werden müssen.

PTP-Move:	<i>erSetTargetLocation_Move_Joint (tarloc_hnd, CartPosvec)</i>
JointAbs-Move:	<i>erSetTargetLocation_Move_JointAbs (tarloc_hnd, JointPos, speed_percent)</i>
LIN-Move:	<i>erSetTargetLocation_Move_CIRC (tarloc_hnd, CartPosVec)</i>
CIRC-Move:	<i>erSetTargetLocation_Move_LIN (tarloc_hnd, CartPosVecVia, CartPosVec)</i>

Ist der ToolPath durch die Host-Applikation vollständig definiert genügt ein einziger Aufruf der Methode *erGET_NEXT_TOOLPATH_STEP()* um die Interpolation entlang des gesamten ToolPaths zu starten. In jedem Interpolationstakt können sämtliche Zustände des Roboters mit seinen synchronisierten Geräten (Positioner, Tracking Achse) abgegriffen und beispielsweise in einer 3D-Szene dargestellt werden.

Die Vorteile liegen auf der Hand.

Bisher wurden im ERK über die Methoden *SET_NEXT_TARGET* und *GET_NEXT_STEP* die Zielpositionen jeweils einzeln von der Host-Applikation übergeben und entsprechend vom ERK einzeln verarbeitet. Der Verwaltungsaufwand insbesondere bei Übergängen zum nächsten Target ("need more data", "target reached") war aufwendig. Das führte zu einem erhöhten Integrationsaufwand des Kernels und einer erhöhten Komplexität bei der Implementierung.

Zum anderen wurde auch ein gewisses Robotik Know-how vorausgesetzt, da verschiedene Attribute für die Zielpositionen zur korrekten Berechnung durch den ERK ausgewählt werden musste.

Mit der neuen ToolPath-Funktionalität kann nicht nur eine einzelne Roboterzielposition sondern sofort eine ganze Trajektorie (=ToolPath) an den ERK übergeben werden. Die Achswinkel und weitere Ergebnisse (Achsgeschwindigkeiten, Positionen des TCPs, TCP Geschwindigkeit, Bahnlänge bis zum Ziel, etc.) werden nun für die Interpolation mit einem einzigen Aufruf der Methode `erGET_NEXT_TOOLPATH_STEP()` für alle enthaltenen Zielpositionen in einem Durchlauf berechnet und geschlossen wieder zurückgegeben. So entsteht auf Kundenseite bei der Integration des ERKs in der eigenen Host-Applikation ein merklich reduzierter Aufwand.

Zusätzlicher Nutzen

Weiterhin werden in jedem einzelnen Interpolationsschritt die Roboter-Konfiguration und die Achswerte des Roboters berechnet und im Target abgelegt. Insbesondere die neue Funktion "API-Post-Process" profitiert davon. Sie kann den Content des gesamten ToolPaths in ein Roboterprogramm umsetzen. Der ToolPath bildet so eine neutrale Ebene aus der sich für jede Robotersteuerung ein Roboterprogramm in geeigneter Syntax erzeugen lässt. Auch die neue Funktion "**ERK-ToolBox**" greift auf den ToolPath zu und hat so die Möglichkeit externe Achsen gemäß vorgegebener Constraint-Regeln zu berechnen.

Auch ist es möglich sofort mehrere Trajektorien zu übergeben und der ERK berechnet alle benötigten Werte. Die Bahnreihenfolge wiederum ist manipulierbar. So können einzelne Bahnen de- und wieder aktiviert werden.

Abwärtskompatibilität

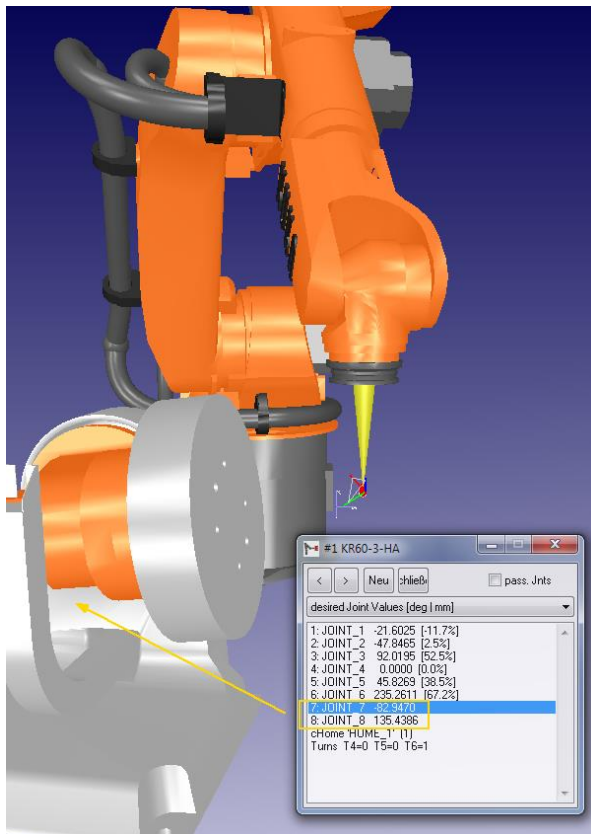
Selbstverständlich werden alle Implementierungen die nicht den ToolPath verwenden weiterhin zu 100% unterstützt und auch zukünftig weitergepflegt. ToolPath bietet lediglich eine Vereinfachung an und schafft neue Möglichkeiten und Funktionen, die den etwas weniger versierten Robotics User unterstützen sollen.

ToolBox - ein Leistungsstarke Ergänzung

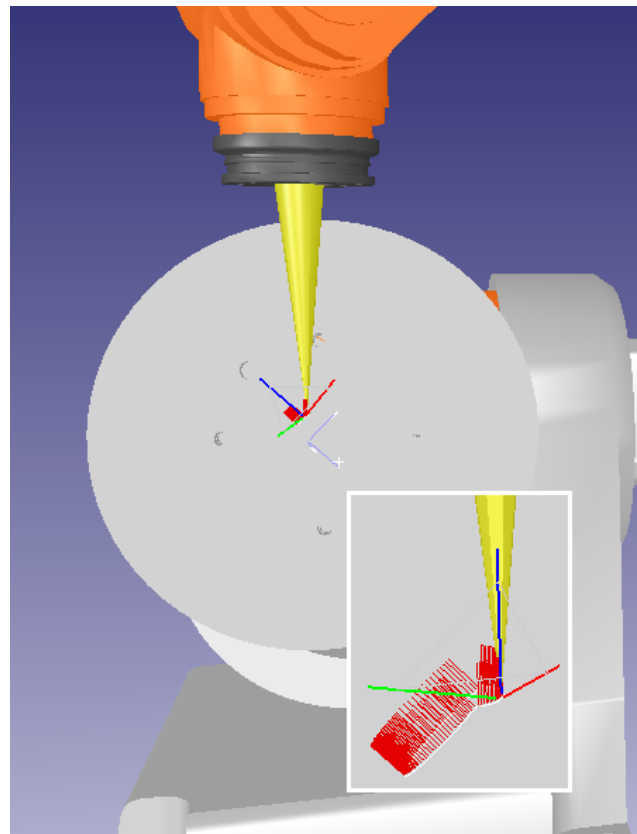
Mit der neuen "ERK-ToolBox"-Funktionalität wird eine leistungsfähige Ergänzung zum EASY-ROB™ Robotics Simulation Kernel angeboten.

Basierend auf einen definierten ToolPath können mittels ERK-ToolBox verschiedenste Berechnungen durchgeführt werden.

In der aktuellen Funktionalität werden der externen Achswerte für einen ein- oder zwei-achsigen Positionierer in Abhängigkeit von vorgegebene Einschränkungen (Constraints) berechnet. Ein typischer Constraint ist die Vorgabe der Werkzeugorientierung bzgl. Bauteiloberfläche, so dass diese beispielsweise immer senkrecht ist, siehe Bild.



Roboter mit einem zwei-achsigen Positionierer



Constraint: Senkrecht zur Bauteiloberfläche

Bei der Berechnung wird automatisch auf die aktuelle Anlagen-Konfiguration (Position und Verknüpfung von Industrierobotern, Lineareinheit und externen Positionierern) zugegriffen.

Vorteile für den Benutzer

- Komplexe Berechnungen werden vom Robotics Simulation Kernel respektive ToolBox übernommen.
- ToolBox stellt eine leistungsstarke Lösung anstelle einer kostenintensiven Eigenentwicklung parat.
- Auf verschiedenste Anforderungen seitens Anwender können OEM-Kunden flexible reagieren (Customization).
- Die ToolBox-Funktionalität ist in einer DLL "*EasySimKernel_tboxx64.dll*" ausgelagert, so dass Endkunden individuell beliefert werden können.
- ToolBox ist skalierbar
- Ihr EASY-ROB Team steht für Ihre individuellen Anforderungen gerne zur Verfügung

Der Aufruf der ToolBox-Berechnung erfolgt aus der Methodenklasse: ERK_CAPI_TOOLPATH_TOOLBOX mit *erTPth_TBoxFct()*. Hierbei werden mit dem ToolPath Handle (ER_TOOLPATH_HND) die gewünschten Constraints übergeben.

◆ erTPth_TBox_Fct()

```
static DLLAPI long ER_STDCALL ERK_CAPI_TOOLPATH_TOOLBOX::erTPth_TBox_Fct ( int          FctIdx,
                                                                    int          FctSubIdx,
                                                                    ER_TOOLPATH_HND er_tpth_hnd,
                                                                    int          constraint_param,
                                                                    char *        svalues = NULL
                                                                    )
```

static

Method for miscellaneous and customized tool path calculations.
Requires DLL EasySimKernel_tboxx64.dll.

Parameters

[in] FctIdx	main function idx = [1..]
[in] FctSubIdx	sub function idx = [1..] corresponding to main function idx
[in] er_tpth_hnd	unique tool path handle ER_TOOLPATH_HND
[in] constraint_param	constraint parameter is an individual bitwise-inclusive-OR operator ()
[in] svalues	string value containing individual input values corresponding to constraint_param

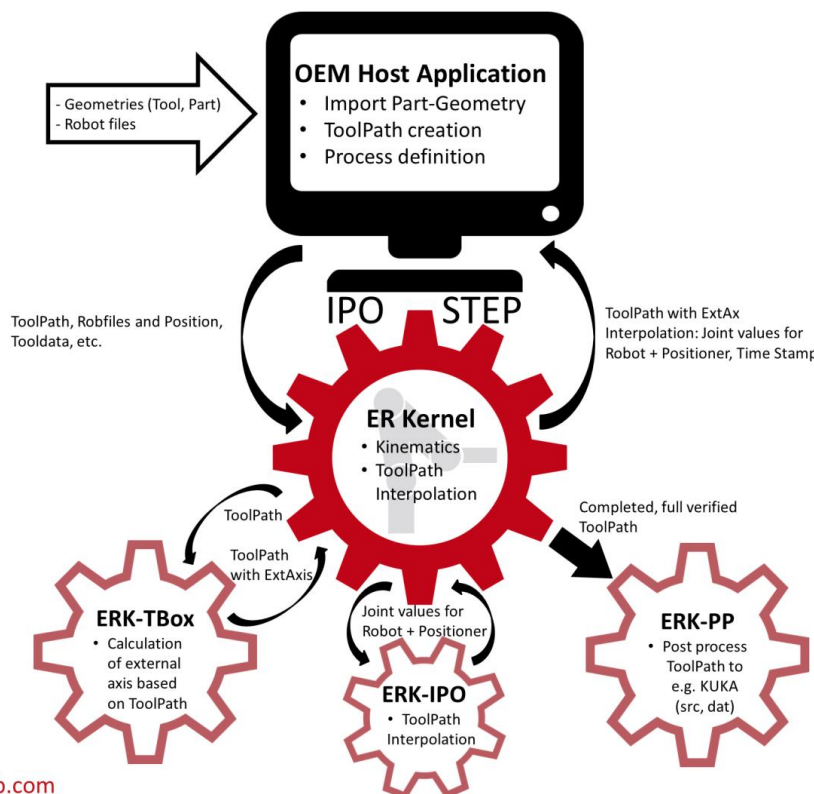
Return values

0 - OK
1 - Error, cannot performe task

Schematische Darstellung: ERK Workflow mit Zusatz APIs und Optionen

Zum besseren Verständnis und Überblick für das Zusammenspiel der Module soll folgender Workflow beitragen.

Workflow: ERK with HostApplication



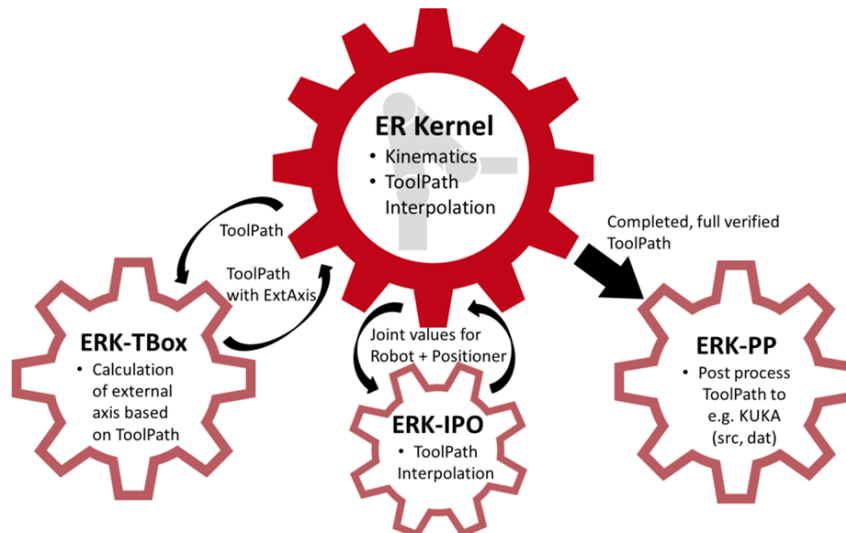
www.easy-rob.com

Typischer Ablauf aus der Host-Applikation heraus

1. Die Host-Applikation lädt die Robot files, positioniert und verknüpft diese
2. Die Host-Applikation erzeugt/erneuert einen ToolPath inklusive Prozessdefinitionen
3. Aufruf von **ERK-ToolBox** zur Berechnung der externen Achsen (*erTPth_TBoxFct()*).
4. Starten der Interpolation mit **ERK-IPO** (*erGET_NEXT_TOOLPATH_STEP()*)
Parallele Visualisierung in jedem Interpolationsschritt, mit gleichzeitiger Kollisionserkennung.
5. Verifikation des Roboterbewegung; Ziel: Fehlerfreier ToolPath, evtl. Anpassung, Optimierung → 2
6. Roboterprogramm erzeugen, Aufruf von **ERK-PP** (API-Post-Process - *erTPth_PostProc()*)

API-Post-Process - Roboterprogramme erzeugen

Eine weitere leistungsstarke Erweiterung zum EASY-ROB™ Robotics Simulation Kernel ist die API-Post-Process (**ERK-PP**) um Roboterprogramme zu erzeugen. Die Basis-Funktionalität wurde vom EASY-ROB™ Robotics Simulator Professional und dem EASY-ROB™ Robotics Framework übernommen.



Um Unterschied dazu greift der ERK-PP auf den ToolPath zu und setzt die einzelnen Anweisungen in ein Roboterprogramm um. Zu bemerken ist an dieser Stelle, dass der ToolPath nach einer erfolgreichen Interpolation (**ERK-IPO**) alle nötigen Daten enthält um ein Zielprogramm zu erzeugen. Dazu zählen beispielsweise die Konfiguration und Achswinkelstellung des Roboters in jeder Zielposition (Target), die benötigt werden um die "Signs" und "Turns" richtig herauszuschreiben. Weiterhin beinhaltet jedes Target auch individuelle Instruktionen, was z.B. native prozessabhängige Anweisungen sein können.

Mit dem erfolgreichen Konzept eines **API** kann der OEM-Partner (Hersteller der Host-Applikation) das erzeugte Roboterprogramm individuell auf den Prozess und Robotersteuerung anpassen, was größtmögliche Flexibilität bietet. Das Prozess Know-how bleibt so stets beim Hersteller.

Vorteile:

- Roboterprogramme individuell erzeugen, anpassen und erweitern
- Die Anzahl der möglichen Post-Prozessoren ist unbegrenzt
- Überschaubares Post-Prozessorbeispiel für KUKA KRL macht individuelle Anpassungen auch für andere Sprachen (**Fanuc/LS, ABB/Rapid, Universal Robots/ URScript, Yaskawa/InFormII, Comau/PDL2**) leicht möglich.
- Prozess-Know-how verbleibt beim Hersteller
- Programmierbeispiele vermeiden kosten- und zeitaufwändige Neuentwicklungen
- Ihr EASY-ROB-Team steht für Ihre Unterstützung und individuelle Anforderungen zur Verfügung

Der Aufruf des Post-Prozessors erfolgt aus der Methodenklasse: ERK_CAPI_TOOLPATH_APIPP mit *erTPth_PostProc()*. Hierbei werden mit dem ToolPath Handle (ER_TOOLPATH_HND) weitere Einstellungen übergeben, wie z.B. Name der erzeugten Roboterprogrammdatei und Zielverzeichnis. Die Parameter *FctIdx* und *FctSubIdx* können verwendet werden um die verschiedenen Postprozessoren in der DLL aufzurufen.

◆ erTPth_PostProc()

```
static DLLAPI long ERK_STDCALL ERK_CAPI_TOOLPATH_APIPP::erTPth_PostProc ( char *
                                                                    ApiPP_Dll_Name,
                                                                    int
                                                                    FctIdx,
                                                                    int
                                                                    FctSubIdx,
                                                                    ER_TOOLPATH_HND er_tpth_hnd,
                                                                    char *
                                                                    program_name,
                                                                    char *
                                                                    target_path = NULL,
                                                                    int
                                                                    pp_param = 0,
                                                                    char *
                                                                    svalues = NULL
                                                                    )
```

static

Method for post processing, creating a robot program for a tool path

An example Visual Studio Project is available, creating a robot program for KUKA Controller.

The name of the DLL is user defined, e.g. 'EasySimKernel_apippx64.dll'

Remarks

The DLL is loaded and linked will calling this method. Thus, the DLL can be generated again without restarting the Host application.

This allows quick changes and adjustments in the shop floor.

```
// Example:
ER_TOOLPATH_HND my_tp_hnd = er_toolpath_hnds[0];           // the one to work with
// Calling PostProcess
char *ApiPP_Dll_Name = {"EasySimKernel_apippx64.dll"}; // ApiPP_Dll_Name Name of ERK_APIPP DLL
int pp_FctIdx = 1; // FctIdx main function idx = [1..]
int pp_FctSubIdx = 1; // FctSubIdx sub function idx = [1..] corresponding to main function idx
char *program_name = {"Kuka-KR60-3-HA"}; // program_name Robot program name without extension
char *target_path = NULL; // target_path target path of created robot program
int pp_param = 0; // pp_param individual input value
char *pp_svalues = {"0 0 0"}; // svalues string value containing individual input values corresponding to pp_param
int res = erk_toolpath_apipp.erTPth_PostProc(ApiPP_Dll_Name,pp_FctIdx,pp_FctSubIdx,my_tp_hnd, program_name,target_path,pp_param,pp_svalues);
if(!ret)
    return 1; // failed
// success, continue
...
```

Parameters

[in] ApiPP_Dll_Name	Name of ERK_APIPP DLL
[in] FctIdx	main function idx = [1..]
[in] FctSubIdx	sub function idx = [1..] corresponding to main function idx
[in] er_tpth_hnd	unique tool path handle ER_TOOLPATH_HND
[in] program_name	Robot program name without extension
[in] target_path	target path of created robot program
[in] pp_param	individual input value
[in] svalues	string value containing individual input values corresponding to pp_param

Return values

0 - OK
1 - Error, cannot perform task

Zu guter Letzt - der ERK-PP wird dynamisch geladen

Beim Aufruf des Post-Prozessors wird die DLL "EasySimKernel_apippx64.dll" dynamisch geladen und am Schluss wieder freigegeben. Somit kann während der Arbeit in der Host-Applikation der ERK-PP angepasst werden ohne die Host-Applikation neu zu starten, die Scene zu laden und alle weiteren Einstellungen vornehmen zu müssen. Das spart erheblich Zeit bei der Inbetriebnahme vor Ort.

Technische Änderungen und Verbesserungen sind vorbehalten

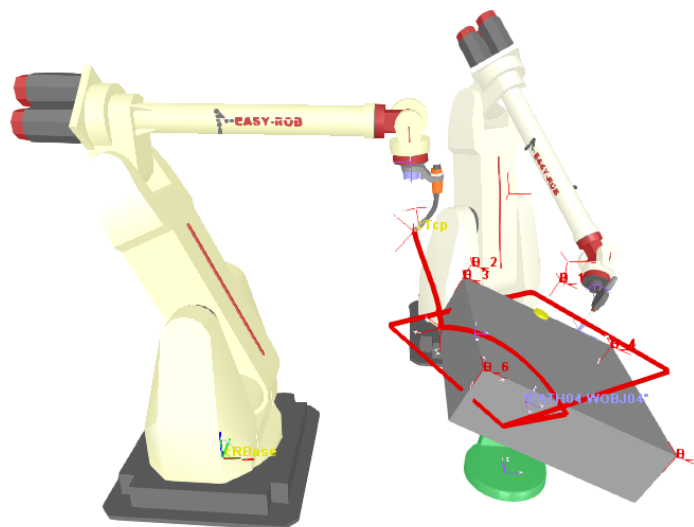
14/36

AutoPath™ als eigenständiges Modul verfügbar

Die AutoPath™ Funktionalität für die kollisionsfreie Bahnplanung steht jetzt neben EASY-ROB™ Robotics Simulator Professional (Single- und Multi-Robot) und dem Robotics Simulation Framework auch dem EASY-ROB™ Robotics Simulation Kernel (ERK) als eigenständiges Modul zur Verfügung.

Damit verfolgt die EASY-ROB Software GmbH weiter Ihr Ziel, eine große, stets passende Familie von harmonisierten Software Modulen für die Robotersimulation anbieten zu können. Für alle EASY-ROB™ Kunden bedeutet das, für alle Anforderungen die in Zukunft auf Sie zukommen, bestens eingestellt zu sein.

Das AutoPath™ Modul hat als Zielgruppe ganz klar alle Kunden des ERKs im Visier, die neben der verlässlichen Robotik-Simulation die Arbeit des Bedieners erheblich vereinfachen wollen. Oder auch alle interessierten Firmen, die eine eigene Bewegungsplanung und –ausführung bereits einsetzen und einen großen Schritt Richtung *Next Level Robot Programming* machen wollen, bisher aber eine solche kostenintensive Eigenentwicklung gescheut haben.



Beispiel-Screenshot einer Multi-Robot Anwendung

Callback-Funktion

Nach der erfolgreichen Trennung von Algorithmus und GUI wurden zusätzliche Funktionen dem AutoPath™ spendiert. Über eine Callback-Funktion kann AutoPath™ intelligent von der Host-Applikation (Ihre Software Anwendung) gesteuert werden:

es werden Start- und Endpositionen vorgegeben und das berechnete Ergebnis sind die jeweiligen Zwischenpositionen (WayPoints), um von der Start- zur Endposition kollisionsfrei zu gelangen.

Auf die Berechnung der Zwischenpositionen kann nun über eine Callback-Funktion Einfluss genommen werden. So kann z.B. ein kartesischer Arbeitsraum vorgegeben werden, in welchem sich die TCP-Position des Roboters bewegen soll. Nachdem diese geometrischen Bedingungen u.a. Achskonfigurationen über die Callback-Funktion erfüllt sind, erfolgt anschließend die Kollisionsprüfung, falls gewünscht auch über EASY-ROB™ Module.

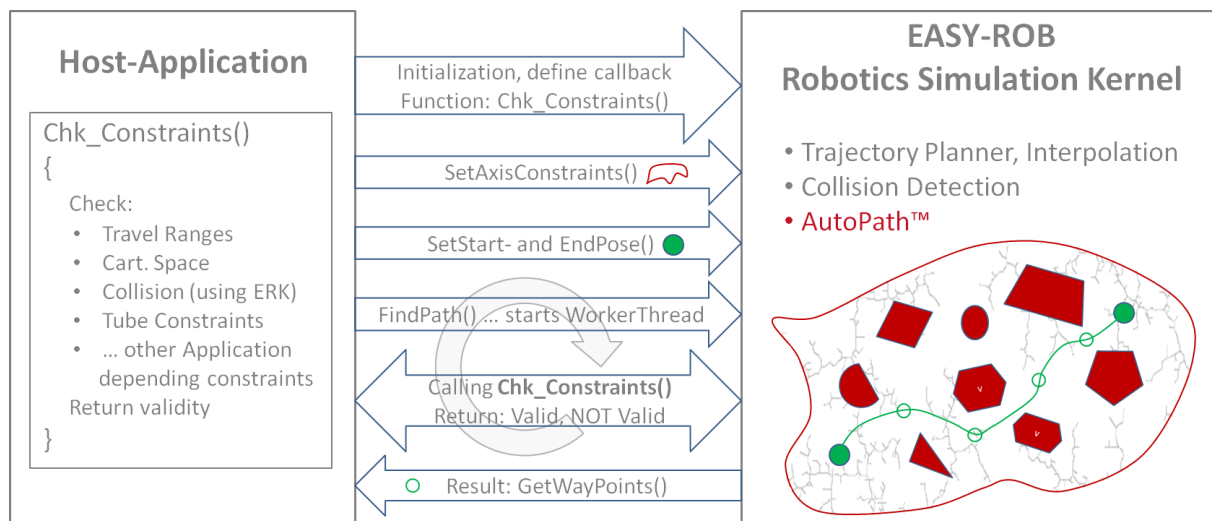
Direkter Bezug zur eigenen Anwendung

Über die Callback-Funktion kann ein direkter Bezug zur Robotik hergestellt werden. Die Zwischenpositionen lassen sich so im Kontext der eigenen Anwendung berechnen und fügen sich entsprechend schnell in Ihre Software ein.

Erfahren Sie auf der nächsten Seite welche es sind und was diese bewirken. Natürlich wird AutoPath™ über eine leistungsfähige API angesteuert, das per Doxygen Dokumentation beschrieben ist.

Für Rückfragen oder weitere benötigte Informationen steht Ihnen unser Vertrieb unter sales@easy-rob.com gerne zur Verfügung.

Kommunikation: Host-Applikation - ERK mit AutoPath™



Ablauf

- **Initialisierung:**
Definition der Callback-Funktion und der Verfahrbereiche der Roboter/Kinematiken.
Die Callback-Funktion, die Bestandteil der Host-Applikation ist, definiert die Constraint-Bedingungen, die je nach technischer Anforderung und Priorität geprüft werden.
- **Setzen der Start- und Zielposition, welche die Constraint-Bedingungen erfüllen müssen.**
- **FindPath:**
Starten des AutoPath™-Algorithmus im Worker Thread, zyklisches aufrufen der Callback-Funktion zur Prüfung der Constraint-Bedingungen. → Return: Valid or NOT Valid
- **Ergebnis → WayPoints**
Wurde ein kollisionsfreier Weg gefunden, der alle Constraint-Bedingungen aus der Callback-Funktion erfüllt, kann eine Liste mit WayPoints abgerufen werden. WayPoints sind achsspezifische Zwischen-Punkte die mit vollsynchronem PTP interpoliert werden müssen.

Vorteile und Anwendungsmöglichkeiten

Vorteile

- Automatische Berechnung von TagPoints und Achskonfigurationen
- Callback Funktionen
- Leichte Integration in technologiebasierte Softwarelösungen
- API Verfügbar

Anwendungsmöglichkeiten

- Industrieroboter
- Serviceroboter
- Animation und Simulation
- Bewegungsplanung
- Montagetests
- Offline Programmierung
- Messprotokolle
- Autonomes Fahren
- Branchenunabhängig

Feature-Übersicht

Automatische Berechnung von TagPoints

- Hindernisse werden durch automatische Berechnung von TagPoints
- mittels ausgereiftem RRT Suchalgorithmus umgangen



Callback Funktionen

- Cartesian Space
- Collision
- Tube Constraints
- Travel Ranges
- Lassen Sie auch Ihre individuellen benötigten Einschränkungen (Constraints) einfließen



Berechnung

Achskonfiguration

- liefert zu jedem TagPoint auch die Achskonfiguration



Vorgabe Verfahrbereiche

- Axis-Constraints für die Verfahrbereiche können via API definiert werden



API

- C/C++ und C# Methodenklasse ERAuto_CAPI



Integration

- Detaillierte Doxygen Dokumentation
- Programmierbeispiele für MS Visual Studio® C/C++ und C#



Privaten Benutzerschlüssel (OwnerKey) festlegen

Dateien von Arbeitszellen (*.cel), Roboter-Baugruppen (*.ras) und Kinematiken (*.rob) können zwecks besserer und schnellerer Editierbarkeit auch als ASCII Dateien gespeichert werden. So stehen diverse Informationen z.B. über Koordinatensysteme oder Achslängen als Klartext drin. Bei engmaschigen Iterationsschleifen in einer heißen Projektphase ist die Möglichkeit, diese Dateien per Texteditor von außen, d.h. ohne EASY-ROB™, zu ändern, sehr hilfreich.

Encrypten- Verschlüsselung durchführen

Was aber, wenn empfindliche Daten oder Betriebsgeheimnisse am Ende enthalten sind?

Damit Ihre Arbeitszellen sowie Ihre erstellen kinematischen Strukturen vor „fremdem Augen“ geschützt werden, können Sie Ihre erstellten Dateien nun mit dem neuen „usr_ownerkey“ verschlüsseln-encrypten bei EASY-ROB™ genannt.

Per „usr_ownerkey“ kann eine erstellte *.cel-, *.ras- oder *.rob-Datei verschlüsselt und als binäre Datei gespeichert werden. So werden alle enthaltenen Informationen über Ihr Projekt oder Ihre Kinematik für fremde Augen unleserlich gemacht.

Der „usr_ownerkey“ kann individuell festgelegt werden. Damit kann innerhalb einer Firma gesteuert werden, z.B. welcher Benutzer entsprechende Informationen sehen soll und vor allem wird eine Sicherheitslücke beim Austausch der Dateien geschlossen, sollten diese an fremde Firmen oder sogar die Konkurrenz gelangen.

Decrypten- Verschlüsselung rückgängig machen

Natürlich können zuvor verschlüsselte *.cel-, *.ras- oder *.rob-Dateien wieder entschlüsselt werden. Dafür ist einfach nur ihr persönlicher „usr_ownerkey“ nötig und sofort können Ihre Dateien aus EASY-ROB™ heraus wieder als ASCII Dateien abgelegt.

Wo ist der usr_ownerkey zu finden?

Der „usr_ownerkey“ wird in einer *.enc-Datei verschlüsselt gespeichert, z.B. "MyCompany-ownerkey.enc". Der Ort und Name der Datei kann in der Konfigurationsdatei "config.dat" festgelegt und so beim Start von EASY-ROB™ sofort geladen werden. In der Voreinstellung ist es der Benutzerordner:

- „USR_DIR=“, bzw. "USR_OWNERKEY_DIR="

Der Ordner wird standardmäßig im EASY-ROB™ Installationsverzeichnis als Benutzerordner erstellt und festgelegt, kann aber anschließend angepasst bzw. verändert werden.

Weitere Informationen zum Anpassen der Benutzerordner und Pfade finden Sie in der Updatebeschreibung von Version 7.3, Seite 6, Dateiname „Update-ER_v7305-2017_DE.pdf“.

Benutzerschlüssel nur in EASY-ROB™ direkt veränderbar

Damit der „usr_ownerkey“ nicht durch Dritte von außen verändert werden kann, muss dieser über die EASY-ROB™ Bedienoberfläche eingestellt oder geändert werden:

- Menu Aux → Einstellungen → Verschlüsselung → Benutzerschlüssel Definition

Benutzerschlüssel ändern oder neu erzeugen

Bei Änderung eines bestehenden Benutzerschlüssels muss stets zuerst der eingestellte Benutzerschlüssel für eine Verifikation eingegeben werden.

Bei der ersten Verwendung nach erfolgreicher Installation lautet dieser:

- 4711

und greift auf die Benutzerschlüssel-Datei zu:

- usr_ownerkey_4711.enc

Anschließend kann ein neuer Benutzerschlüssel eingegeben werden und in einer neuen Benutzerschlüssel-Datei z.B. "MyCompany-ownerkey.enc" gespeichert werden.

Hinweis: Bitte schreiben Sie Ihren Benutzerschlüssel NICHT in den Benutzerschlüssel-Dateinamen.

Verschlüsselung *ohne* Benutzerschlüssel

Wenn Sie Ihre *.cel-, *.ras- oder *.rob-Dateien *ohne* Benutzerschlüssel verschlüsseln wollen, gehen Sie über die EASY-ROB™-Bedienoberfläche und aktivieren die Verschlüsselung wie folgt:

- Menu Aux → Einstellungen → Verschlüsselung → Datei Verschlüsselung

Hinweis: Obwohl Ihre Datei nun verschlüsselt ist, kann jeder Benutzer mit einer EASY-ROB™ Vollversion Ihre Datei wieder entschlüsseln.

Verschlüsselung *mit* Benutzerschlüssel

Wenn Sie Ihre *.cel-, *.ras- oder *.rob-Dateien *mit* Benutzerschlüssel verschlüsseln wollen, müssen *beide* Check Boxen gesetzt sein:

<input checked="" type="checkbox"/>	Datei Verschlüsselung	ON/OFF
<input checked="" type="checkbox"/>	Benutzerschlüssel-Aktiv	ON/OFF
Benutzerschlüssel Definition		

- Menu Aux → Einstellungen → Verschlüsselung → Datei Verschlüsselung
- und
- Menu Aux → Einstellungen → Verschlüsselung → Benutzerschlüssel Aktiv

3D-PDF Export mit Animation als SDK

Suchen Sie für Ihre Software einen belastbaren 3D-PDF Export mit Animation und sind bisher nicht fündig geworden?

Dann haben wir für Sie genau die richtige Lösung:
Erweitern Sie Ihr Produkt um unseren erfolgreichen 3D-PDF Export!



Anwendungsbereiche 3D-PDF Export SDK

- Schnelle und einfache Präsentation - auch gegenüber Dritten
- Weitergabe von animierten interaktiven Simulations-Konzepten
- Montage- und Wartungsanleitungen
- Dokumentation von erklärungsintensiven Inhalten
- Universelles Schulungs- und Trainingsmaterial
- Interaktive Vertriebsmaterialien für ein verbessertes Produktverständnis bei Kunden

Das EASY-ROB™ 3D-PDF SDK ermöglicht Ihnen, für oben genannten Anwendungsbereiche benötigte Funktionen einfach in Ihre Software zu implementieren.

Bewegungsablauf mit Animation in 3D-PDF speichern

Im Adobe® Reader können Sie mit der Navigationsleiste den aufgezeichneten Bewegungsablauf starten, pausieren, stoppen, vor- und zurückspulen, sowie die Geschwindigkeit (x1/64-fache bis x64-fache) ändern. Die Zeitangabe gibt die dabei die reale Prozesszeit an.

Das Layout ist frei definierbar



Navigationsleiste im Adobe® Reader

Programmierschnittstelle

- Für das 3D-PDF SDK wird eine C/C++ und C# Methodenklasse zur Verfügung gestellt:
ER3DPDF_CAPI

Für Rückfragen oder weitere benötigte Informationen steht Ihnen unser Vertrieb unter sales@easy-rob.com gerne zur Verfügung.

Selbst-Kollision (Itself Collision)


Eine leistungsstarke Kollisionskontrolle zeichnet EASY-ROB™ schon seit vielen Jahren aus. Die Kollisionskontrolle lässt eine sichere Simulation zu und unterscheidet auch innerhalb von feinstrukturierten Assembly Strukturen (Baugruppen) der CAD Geometrien.

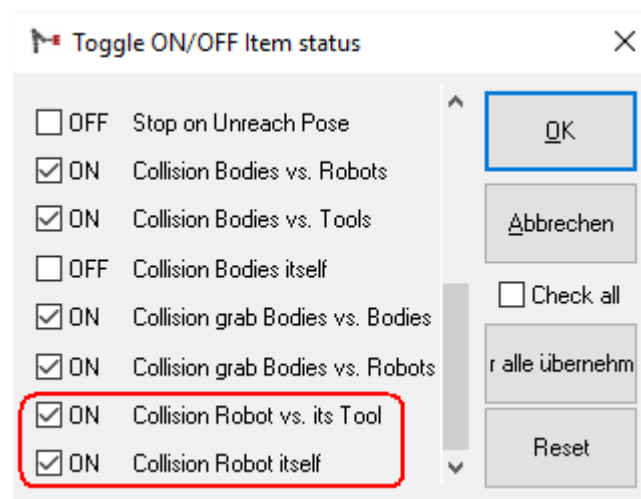
Bisher wurden Kollisionen innerhalb eines Robotermodells nicht erkannt. Trat eine Kollision beispielsweise zwischen Achse 5 und Achse 2 eines Gerätes auf, konnte diese Kollision nicht erkannt und angezeigt werden. In der Regel wurde diese Problematik durch die richtige Einstellung der Verfahrbereiche eingegrenzt.

Letztendlich musste der Benutzer dafür Sorge tragen, dass solche „*Itself Kollisionen*“, also Selbst-Kollisionen, innerhalb der Roboter Kinematik verhindert wurden. Hier war besonders viel Sorgfalt und Zeit nötig, da der gesamte Simulationslauf überprüft und manuell überwacht werden musste. Die „Stop On“ Funktionen, hier im Besonderen die „Stop on Collision“ Funktion haben die Selbst-Kollisionen bisher nicht beachtet.

Selbst-Kollision global aktivieren

Zuerst muss die ständige Bereitschaft in EASY-ROB™ geschaffen werden, damit die Erkennung der Selbst-Kollisionen zum Tragen kommt.

Dazu öffnen Sie das „Stop On“ Menü  und setzen die Checkbox bei „Collision Robot vs. its Tools“ und bei „Collision Robot itself“



Stop On Menü

Bestätigen Sie Ihre Auswahl mit „Ok“.

Damit der „Collision Robot vs. its Tools“- und der „Collision Robot itself“-Parameter für die weitere Benutzung von EASY-ROB™ permanent gesetzt wird, speichern Sie diese bitte kurz erneut in der Umgebungsdatei ab:

- Menu Datei → Speichern → Umgebungsdatei

Sollten Sie die Umgebungsdatei „easy-rob.env“ manuell editieren, suchen Sie nach folgender Stelle:

```
! COLL_ROBOT_TOOL = 1, enables collision between Robot vs. its Tool
COLL_ROBOT_TOOL 1
! COLL_ROBOT_ROBOT = 1, enables collision between Robot itself
COLL_ROBOT_ROBOT 1
```

Dabei nimmt die Variable den Wert 1 für eine aktive Itself Collision an, und Wert 0 für eine deaktivierte.

Kollisionsausschlusslisten für Roboter erzeugen

Für jedes Gerät/Roboter können Kollisionsausschlusslisten automatisch in EASY-ROB™ erzeugt werden. Kollisionsausschlusslisten enthalten sämtliche Geometrien eines Gerätes die in einer "gutartigen" Stellung, z.B. der Homeposition des Roboters, kollidieren.

Sobald diese Listen erstellt und die It Self Collision aktiviert wurden, erkennt von nun an EASY-ROB™ alle Kollisionen der Geometrien innerhalb eines Roboters, die wiederum *nicht* in der Kollisionsausschlussliste stehen.

Um für das aktive Gerät die Kollisionsausschlussliste zu erzeugen öffnen Sie das Kinematics Window [Strg+K] und wählen:

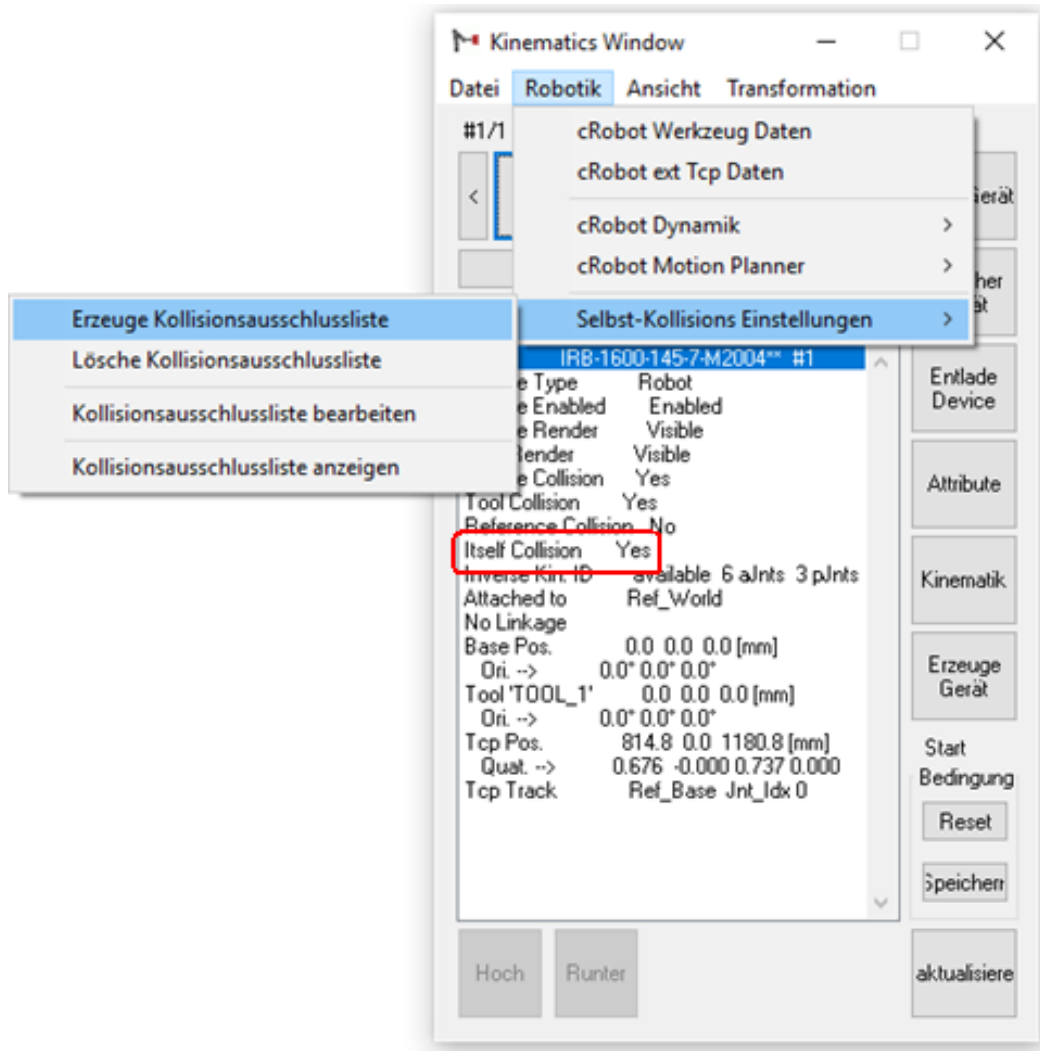
- Menu Robotik → Selbst-Kollisions Einstellungen → Erzeuge Kollisionsausschlussliste

Im nächsten Schritt muss für das aktuelle Gerät die "Itself-Collision" im Kinematics Window eingeschaltet werden:

- Doppel-Klick auf "Itsel Collision No/Yes"

Device Collision	Yes
Tool Collision	Yes
Reference Collision	No
Itself Collision	Yes
Inverse Kin. ID	available 6 aJnts 0 pJnts
Attached to	Ref_World
No Linkage	

Einstellungen im Kinematics Window



Selbst-Kollisions-Einstellungen im Kinematics Window

Hinweis

Die Kollisionsausschlussliste wird einmalig erzeugt und mit dem Robotermodell/Arbeitszelle abgespeichert. Bei einer weiteren Verwendung des Roboters, auch in anderen Projekten, muss die Liste dann nicht erneut angelegt werden.

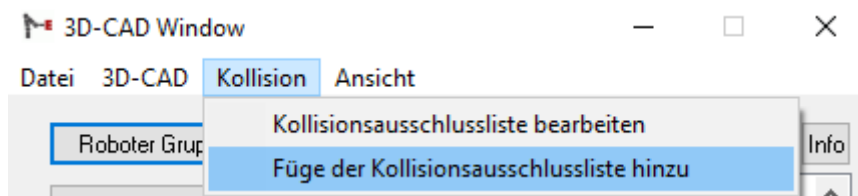
Die Kollisionserkennung arbeitet mit Dreiecken. Diese sind Bestandteil von Geometrien. Auch wenn man Einstellungen im Kinematics Window vornimmt, werden stets Geometrien bei der Betrachtung von Selbst-Kollisionen verwendet.

Individuelle Einstellungen über das 3D-CAD Window

Sollten weitere Geometrien in die Kollisionsausschlussliste aufgenommen werden, geschieht das über das 3D-CAD Window.

Markieren Sie im 3D-CAD Window Geometriepaare bzw. mehrere Geometrien, die der Kollisionsausschlussliste hinzugefügt werden sollen. Wählen Sie anschließend im Menu Kollision

- Menu Kollision → Füge der Kollisionsausschlussliste hinzu.



Individuelles Hinzufügen

Falls das Robotermodell aus einer Vielzahl an Geometriedateien besteht, und das Durchsuchen in der Listenansicht keine Option ist, können Sie die beiden Geometrien auch direkt über die grafische Benutzeroberfläche auswählen.

Dazu klicken Sie den „Pick“ Button an und halten [Strg] gedrückt während Sie die Geometriedateien anwählen. Diese werden anschließend in der Listenansicht „gehighlighted“. Wiederholen Sie jetzt einfach den zuvor erklärten Schritt „Füge der Kollisionsausschlussliste hinzu“.

Natürlich können die Kollisionsausschlusslisten auch editiert werden. Hier wählen Sie bitte:

- 3D-CAD Window → Kollision → Kollisionsausschlussliste bearbeiten.

Bemerkung:

- Die Geometrie-Indizes beginnen mit 1 bis n, mit n - Anzahl der Geometrien in der Gruppe
- Geometrie-Indizes werden durch Leerzeichen getrennt
- Geometrie-Indizes müssen nicht in einer bestimmten Reihenfolge auftreten

Verbesserte Referenz-Kollision

Eine leistungsstarke Kollisionskontrolle zeichnet EASY-ROB™ schon seit vielen Jahren aus. Die Kollisionskontrolle lässt eine sichere Simulation zu und unterscheidet auch innerhalb von feinstrukturierten Assembly Strukturen (Baugruppen) der CAD Geometrien.

Bisher wurden Kollisionen nicht erkannt, wenn z.B. ein Werkzeug als Gerät (Device) an ein Robotermodell direkt „attached“ wurde ("Reference Collision No"). Trat trotzdem eine Kollision auf, z.B. Werkzeug mit den Hand- oder Grundachsen des Roboters, wurde diese Kollision nicht erkannt und auch nicht angezeigt.

Letztendlich musste der Benutzer dafür Sorge tragen, dass solche „Referenz-Kollisionen“, zwischen Werkzeug und Roboter, verhindert wurden.

Ähnlich wie bei der Itself-Collision wird wiederum eine Ausschlussliste angelegt, so dass Geometrien die nach dem Attachen oder Greifen eines Gerätes sowieso kollidieren vom Kollisionstest ausgeschlossen werden.

„Referenz-Kollision“ aktivieren, Geometrien ausschließen

Setzen Sie für das Gerät welches attached oder gegriffen wird (Child device) die Referenz-Kollision zuerst auf "Yes". Dazu öffnen Sie das „Kinematics Window“ und setzen per Doppelklick den Wert auf „Yes“:

■ Kinematics Window → Reference Kollision → Yes

Im nächsten Schritt wählen Sie die Geometrien des Parent devices und des Child devices aus, die in der Grundstellung kollidieren und somit im Kollisionstest ausgeschlossen werden sollen.

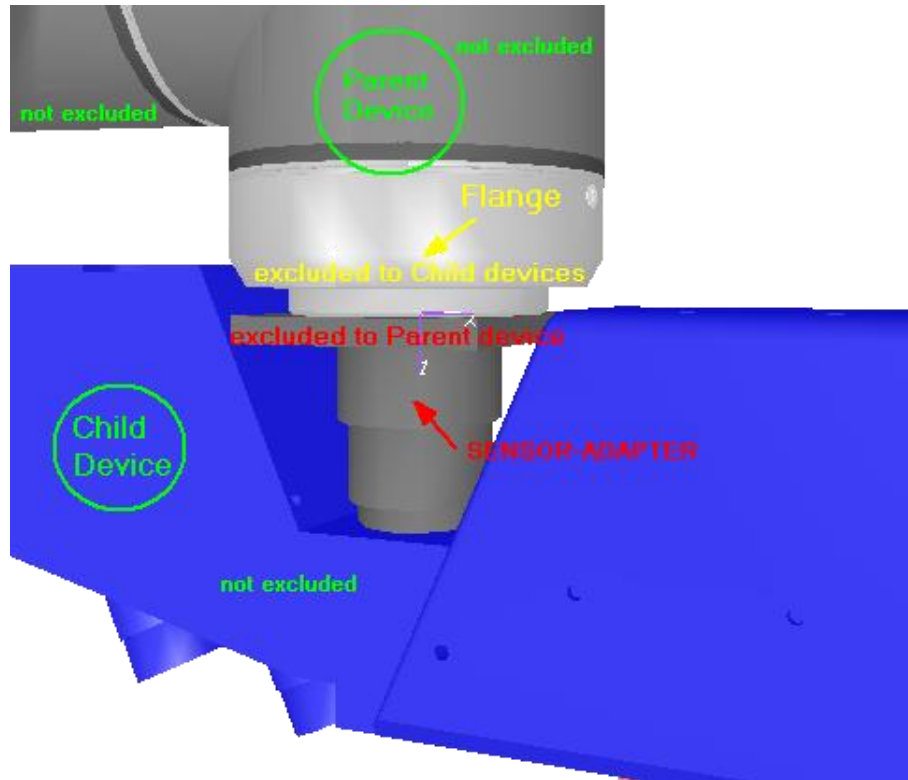
Letztendlich werden bei der Kollisionskontrolle im Allgemeinen und im Speziellen Geometrien überwacht, die zuvor genannten Geräte benutzen diese eigentlich nur. Daher muss man im letzten Schritt eine Abhängigkeit für die Verknüpften oder „attachen“ Geräte auf Geometrieebene festlegen.

Das eine wird nun zum Parent device (Eltern) und das andere wird zum Child device (Kind)-

Ähnliche Bezeichnungen findet man im CAD: hier gibt es Bauteile „Parts“ die von einer Grundgeometrie abstammen und man nennt diesen Zusammenhang ebenfalls „Parent-Child“ Beziehung.

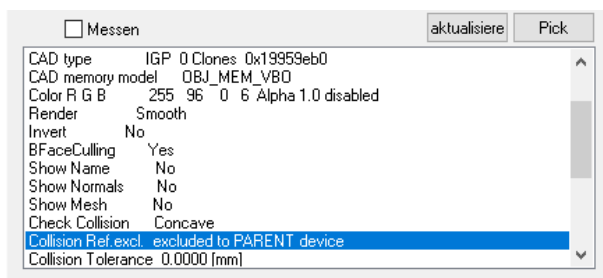
So wird z.B. der Roboter zum Parent device, hier die Geometrie der Achse 6, und die Adapter-Geometrie des Sensors, wird nun zum Child device, siehe bitte nächstes Bild:

Parent- und Child devices

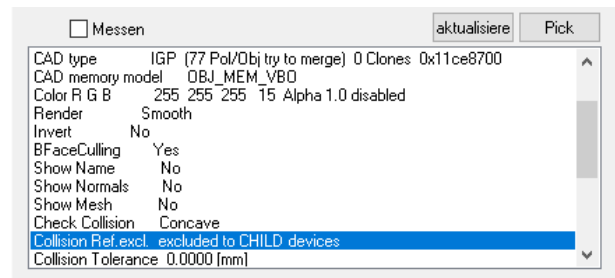


Der Roboter ist das Parent device und der Sensor das Child device
Außer der Flange-Geometrie des Parent devices und der Adapter-Geometrie des Child Devices sollen alle Geometrien vom Parent- und Child device gegeneinander auf Kollision geprüft werden.

Beispiel-Einstellung im 3D-CAD Window für die Adapter-Geometrie des Child devices und Flange Geometrie des Parent Devices



Richtig eingestelltes Child device



Richtig eingestelltes Parent device

Vollständige Roboterbibliotheken

In EASY-ROB™ stehen vollständige Bibliotheken zur Einbindung aller bedeutenden Robotertypen des Marktes bereit. Dazu zählen ABB, b+m, Comau, Denso, Eisenmann, Fanuc, Guedel, igm, Kawasaki, KUKA, Mitsubishi, OTC-Daihen, Reis, Stäubli, Tricept, Unimation, Universal Robots und Yaskawa.

Die Roboterbibliotheken von ABB, KUKA, Comau, Fanuc, Stäubli und Yaskawa sind nahezu vollständig und werden permanent von uns gepflegt.

Derzeit sind mehr als 1000 Roboter, Positionierer und externe Tracking-Achsen verschiedenster Hersteller verfügbar.

Weitere Informationen finden Sie bitte hier:

<http://www.easy-rob.com/produkt/erweiterungen/roboter-bibliotheken.html>

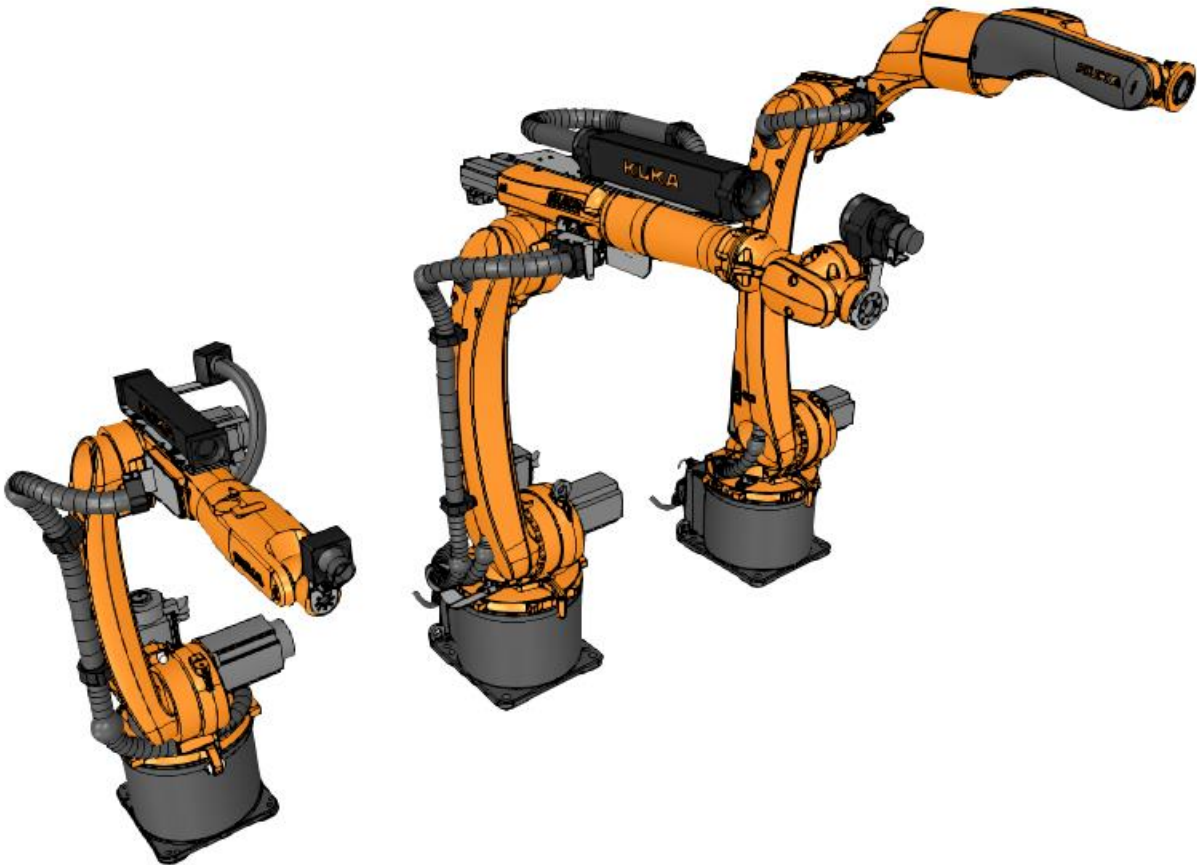
Detailübersicht neue Robotermodelle

Da seit letztem Release besonders viele Roboter hinzugekommen sind, finden Sie eine detaillierte Auflistung der neuen Modelle am Ende der jeweiligen Herstellerseite.

Wichtig:

Nicht vorhandene Roboter, Handlingsysteme, Maschinen, Werkzeuge oder auch spezielle Kinematiken lassen sich in EASY-ROB™ einfach und schnell „virtuell nachbauen“.

KUKA die neue CYBERTECH-Serie



KR-10-R1420-HP

KR-20-R1810

KR-8-R2100-ARC-HW

KUKA

KR-10-R1420
KR-10-R1420-HP
KR-12-R1810
KR-16-R1610
KR-16-R2010
KR-20-R1810
KR-22-R1610

KR-30-3-C
KR-30-3-C-F
KR-3-R540
KR-6-R1820
KR-6-R1820-arc-HW
KR-6-R1820-HP

KR-8-R1420-arc-HW
KR-8-R1620,
KR-8-R1620-arc-HW
KR-8-R1620-HP
KR-8-R2010
KR-8-R2100-arc-HW

Stäubli die neue TX-2 Serie mit CS9 Steuerung



TX2-40

TX2-60

TX2-90

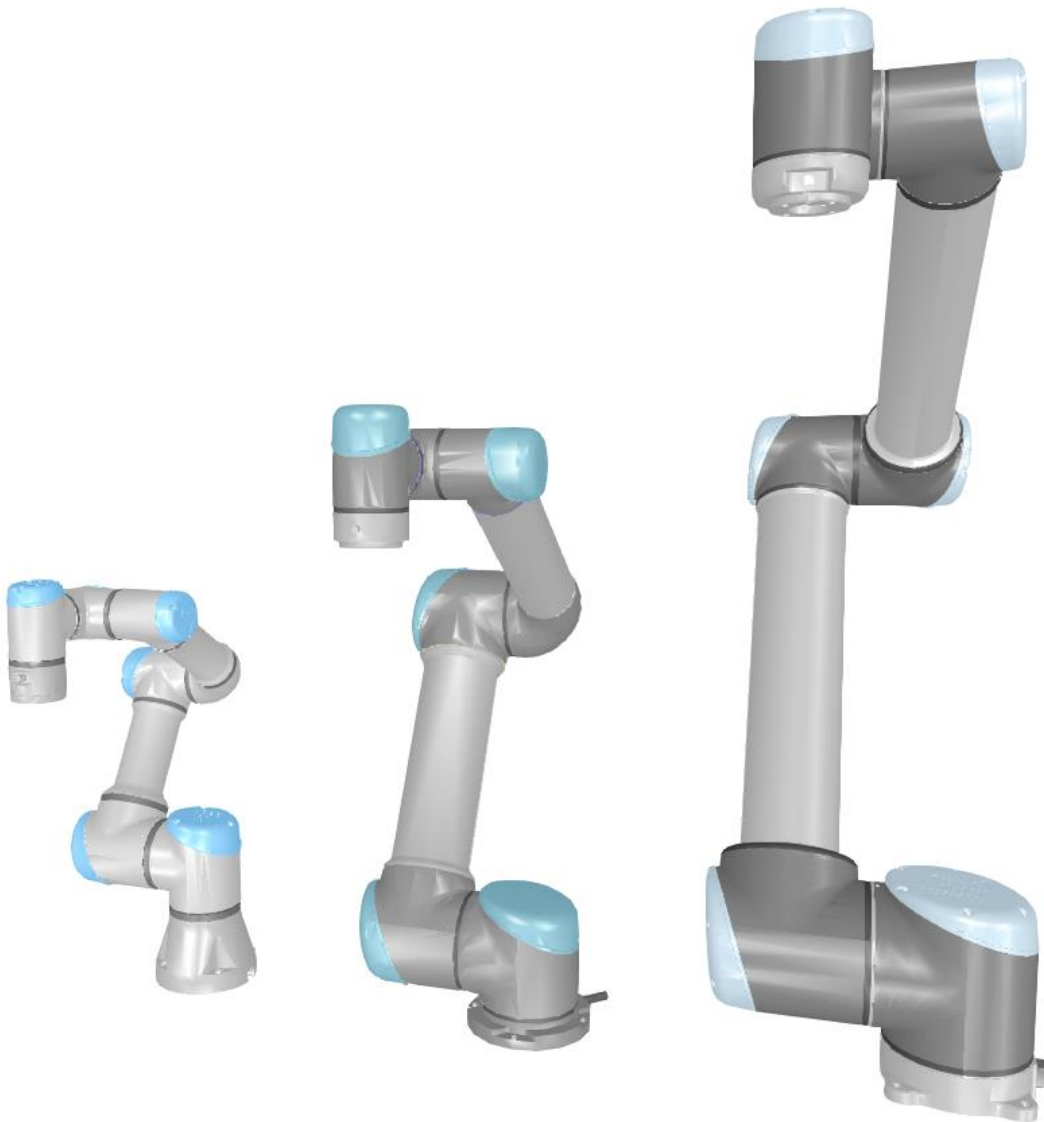
STÄUBLI

TX2-40

TX2-60
TX2-60-L

TX2-90
TX2-90-L
TX2-90-XL

Universal Robots die neue e Serie



 **UNIVERSAL
ROBOTS**

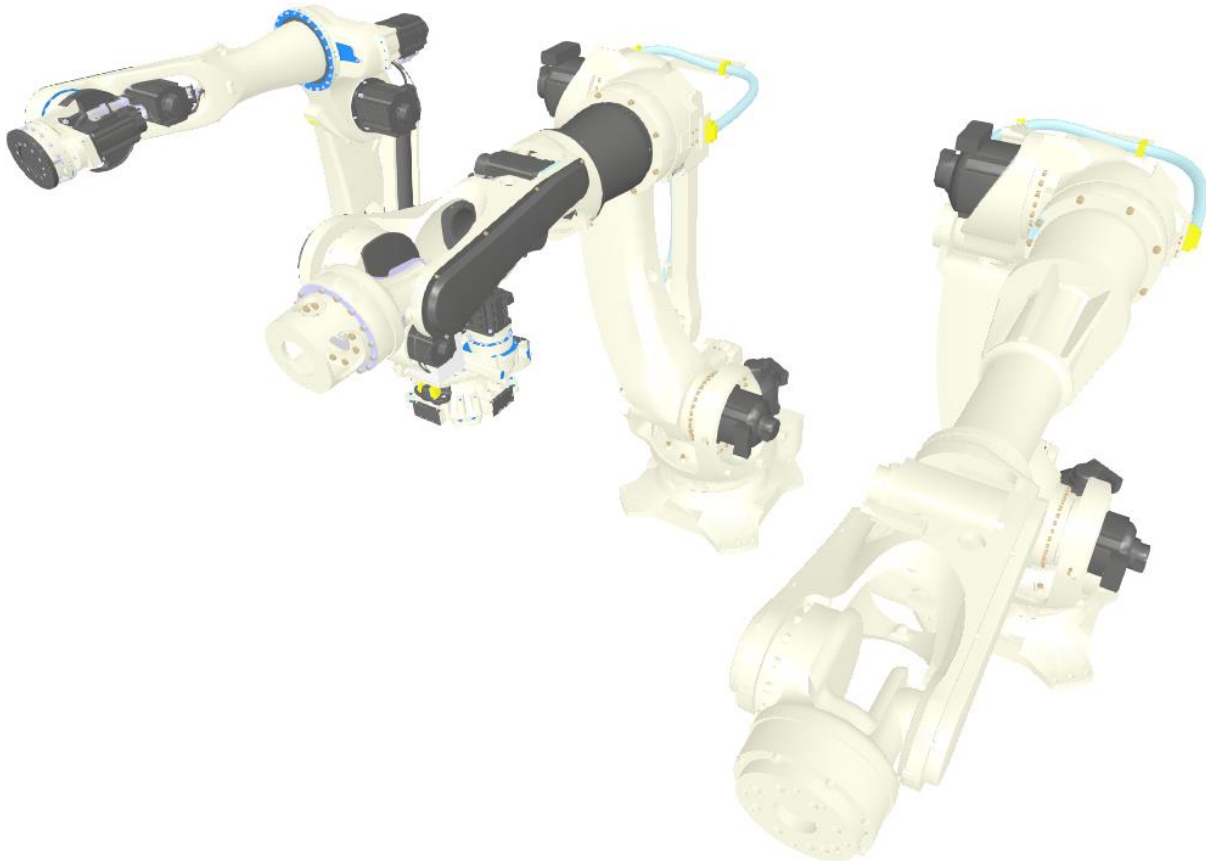
UR3e

UR5e

UR10e

Die bestehenden Serien wurden massiv erweitert:

Kawasaki die CP, CX und BX Serie



CX-110L

BX-300-L

BX-200X

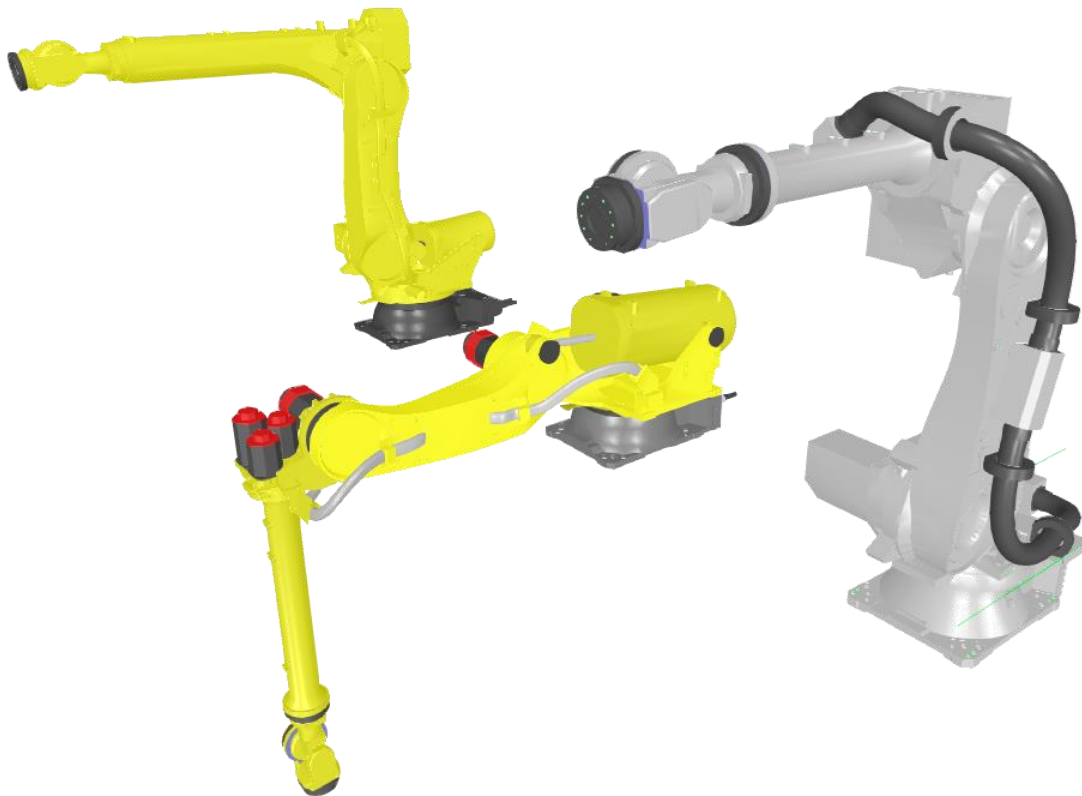


BT-165L
BT-200L
CP180L
CP300L
CP500L
CP700L

BX-100L-C
BX-100S
BX-130X-C
BX-165L-C
BX-165N-C
BX-200L-C
BX-200X
BX-250L
BX-300L

CX-110L
CX-165L
CX-210L
MC004N
MC004V
MS005N

FANUC die R-2000 Serie

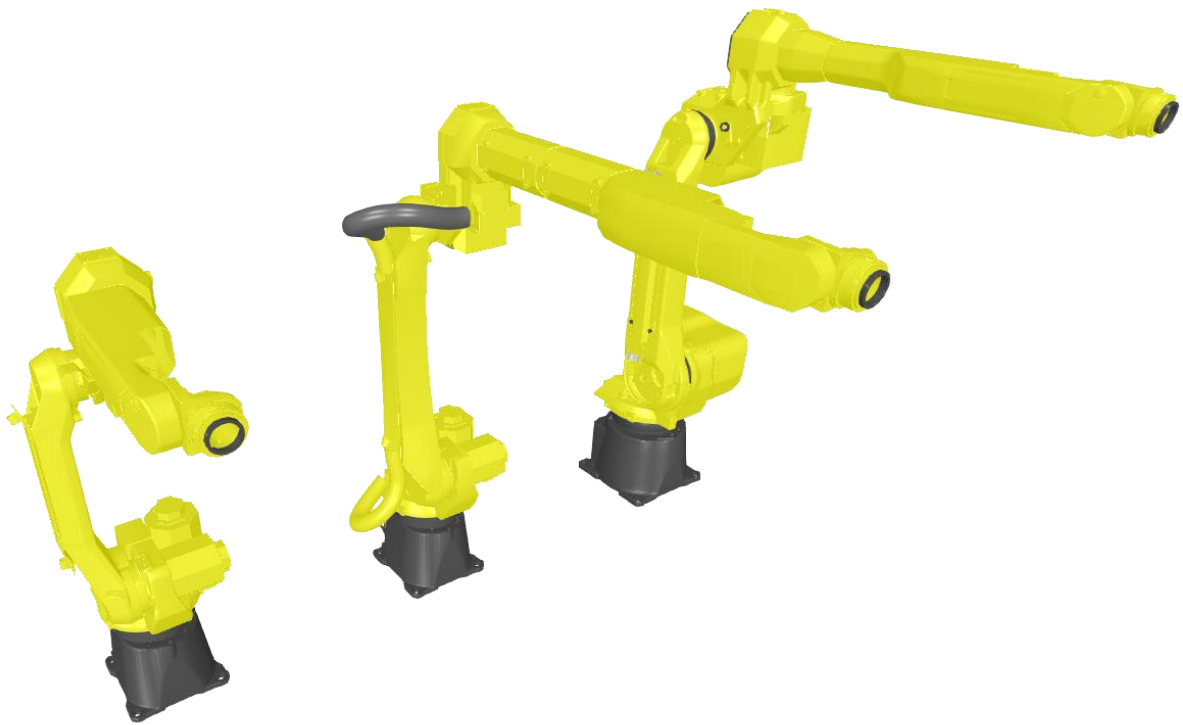


R-2000iC-210L

R-2000iC-100P

R-2000iC-210WE

FANUC die Arc-Mate Serie



Arc-Mate-100-iC-10S

Arc-Mate-100-iC-8L

Arc-Mate-120-iC-12L



ARC-Mate-100iC-12
ARC-Mate-100iC-12S
ARC-Mate-100iC-6L
ARC-Mate-100iC-7L
ARC-Mate-100iC-8L
ARC-Mate-100iCe-6L
ARC-Mate-120iC-12L

M-900iB-280
M-900iB-280L
M-900iB-360
M-900iB-400L
M-900iB-700.rob

R-2000iC-100P
R-2000iC-210L
R-2000iC-210WE
R-2000iC-220U.rob
R-2000iC-270F.rob

Kontakt

EASY-ROB Software GmbH

Adresse: Hauptstr. 42
65719 Hofheim am Taunus
Germany

Kontakt: Stefan Anton, Patryk Lischka

Tel.: +49 (0) 6192 921 70-77 / -79

Fax: +49 (0) 6192 921 70 66

Email: contact@easy-rob.com
sales@easy-rob.com

Web: www.easy-rob.com

Online Shop: <http://www.easy-rob.com/produkt/shop.html/>

EASY-ROB Kundenbereich

Inhalte: Programm-Updates und Roboterbibliotheken

Web: www.easy-rob.com/special/kundenbereich

Zugangsdaten:

Benutzer:	customer
Passwort:	*****

Eigene Notizen