SZKOLENIE Z OBSŁUGI ROBOTÓW STÄUBLI

cz.1 – poziom operator

Seria robotów: TX2/TS2/**CS9**

TABELA ZAGADNIEŃ – OPERATOR

CZĘŚĆ 1 - OPERATOR CS9

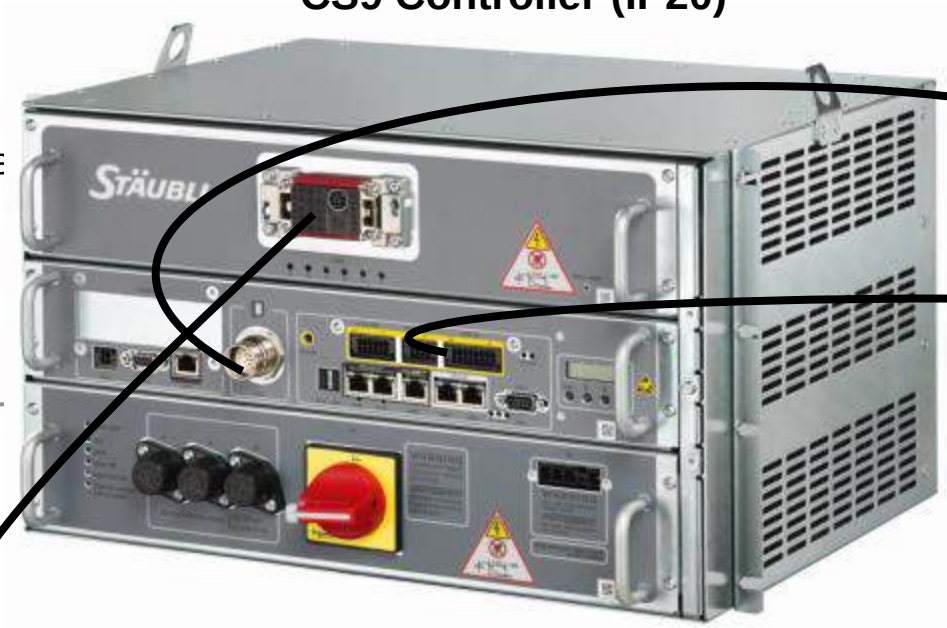
PREZENTACJA SYSTEMU I URUCHOMIENIE ROBOTA

TX2 / CS9: CO W ZESTAWIE

Opcja: SP2 Manual control pendant (MCP) – pilot

Opcja: Podstawa do SP2

CS9 Controller (IP20)



Serwowzmacniacze

Sterownik:

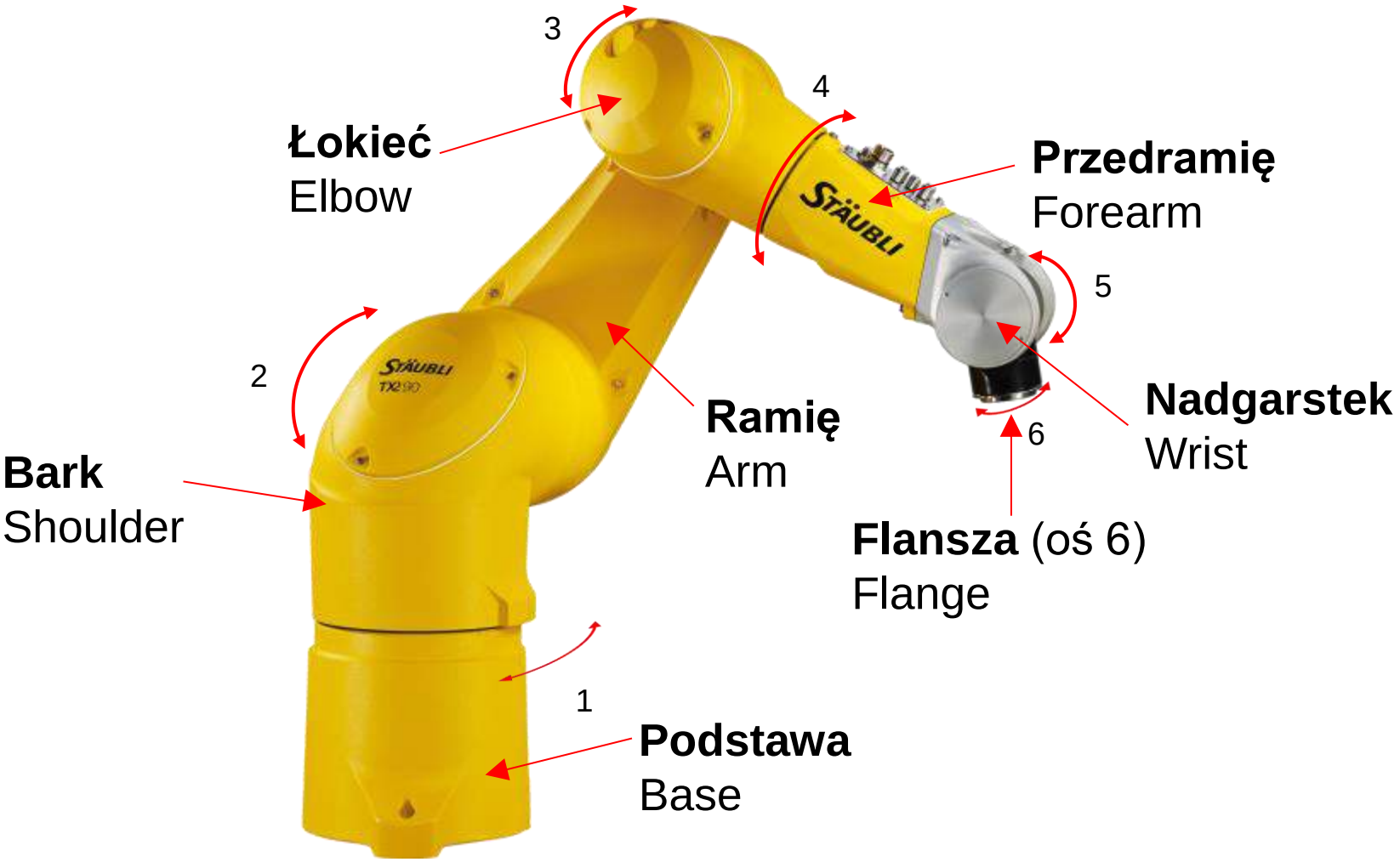
(VAL3) + *Safety*

Zasilacz

Opcja **WMS**:
Working **M**ode **S**election

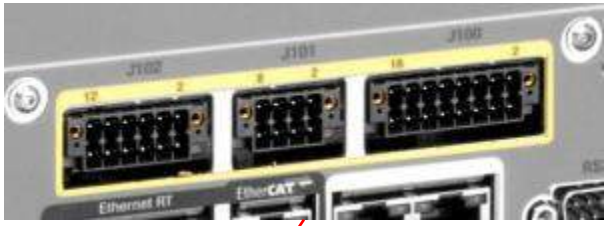
Ramię TX2

TX2 – RAMIĘ

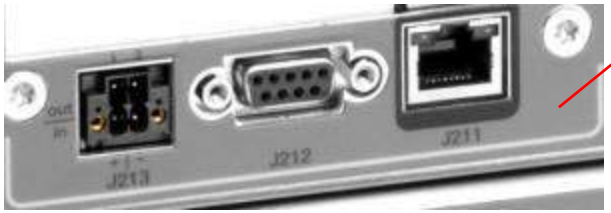

3
4
Łokieć
Elbow
Przedramię
Forearm
5
2
Ramię
Arm
Nadgarstek
Wrist
6
Bark
Shoulder
Flansza (oś 6)
Flange
1
Podstawa
Base

POŁĄCZENIA

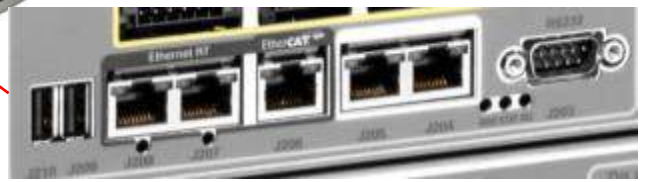
- Sygnały układu bezpieczeństwa (z WMS)



- 1 PCI-express na dodatkowe karty Fieldbus master

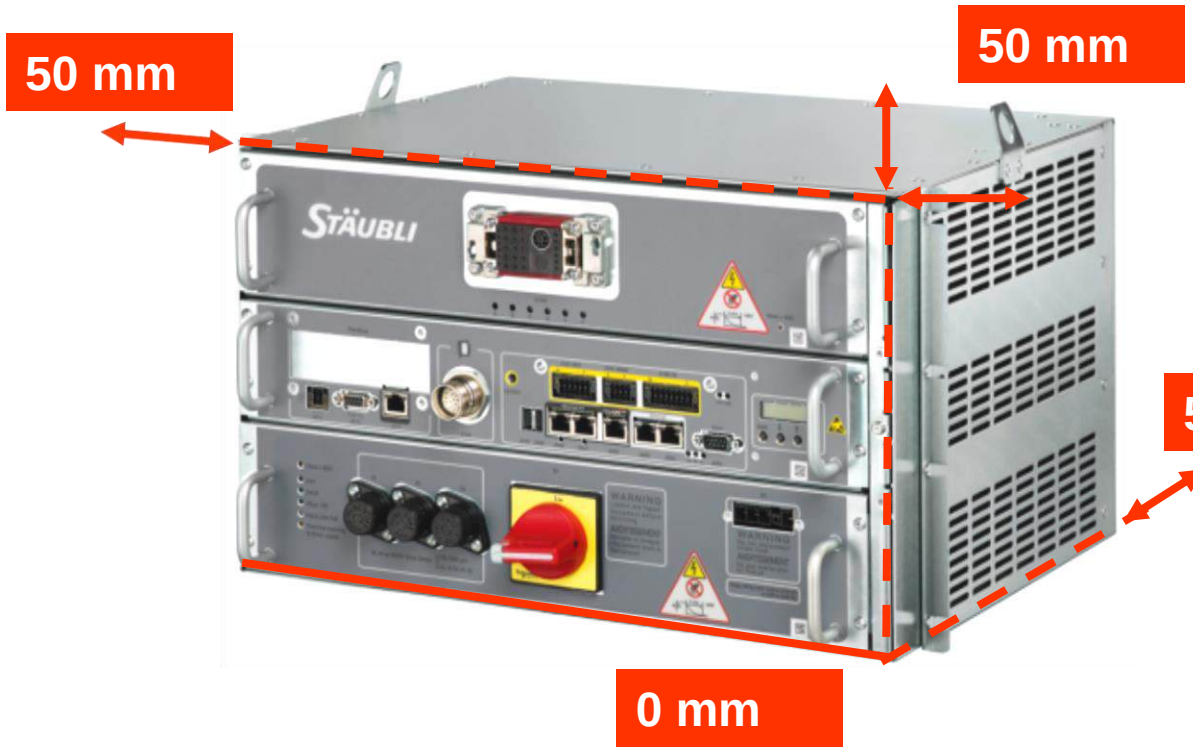


- 1 złącze 24V – zewnętrzne zasilanie układu bezpieczeństwa
- 2 szybkie wejścia
- 2 szybkie wyjścia



- 2 porty Ethernet – Fieldbus slave (1 rodzaj, połączenie daisy chain)
- 1 EtherCAT master
- 2 TCP/IP Ethernet
- 1 RS232

CS9 – IP20 INTEGRACJA



WMS: **W**orking **M**ode **S**election

Przełącznik trybu pracy musi być zainstalowany poza obszarem celi

IP20 – w celu zwiększenia stopnia ochrony IP, kontroler musi być zainstalowany w dodatkowej szafie.

SP2 (IP54), po odłączeniu od kontrolera powinien zostać schowany – nie działa przycisk E-STOP !

SP2 TEACH PENDANT

Przyciski sterowania ruchami robota:

- Przycisk E-Stop
- Przycisk zał./wył. serwonapędy
- Przycisk Move/Hold
- Przyciski zmiany prędkości
- Przyciski ruchu osi
- Przycisk zezwolenia na ruch – „Deadman”

Przyciski nawigacyjne:

- Strona domowa
- Krok wstecz
- Okno kontekstowe

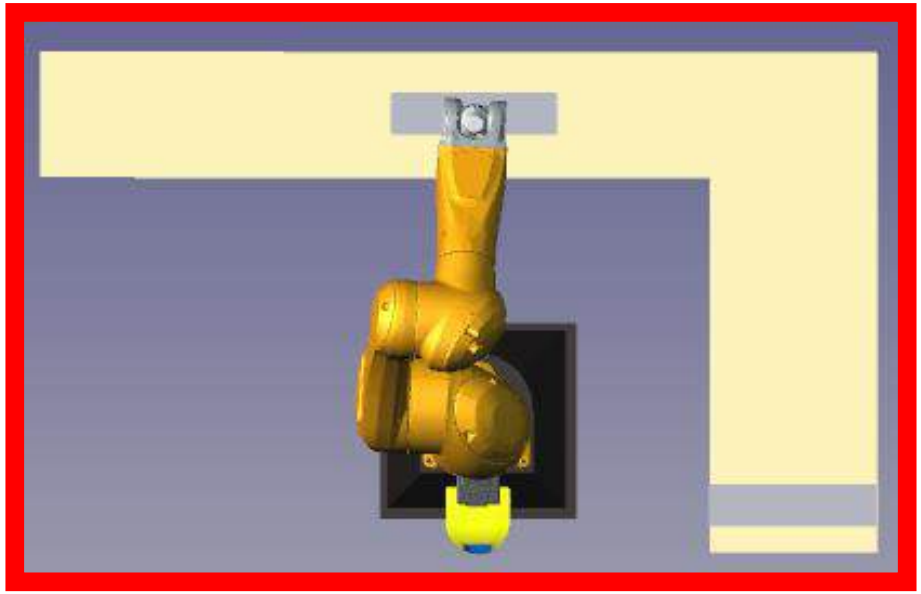
Przyciski skrótów sygnałów **Do**

Port **USB**

ZAAWANSOWANY UKŁADU BEZPIECZEŃSTWA

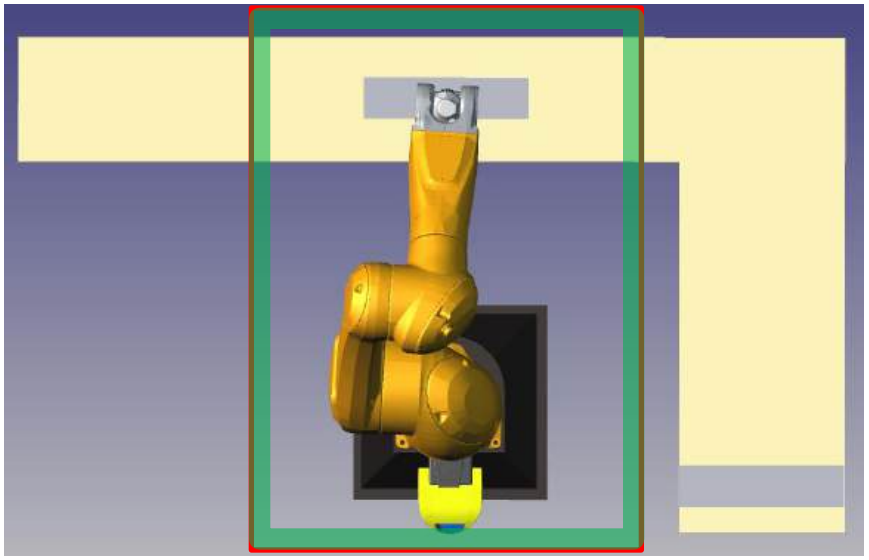
- Ochrona ludzi
- Zmniejszenie obszaru wygrodzonego
- Zmniejszenie obszaru pracy robota

Dotychczasowe rozwiązania



Mocne wygrodzenia

Zaawansowany układ bezpieczeństwa



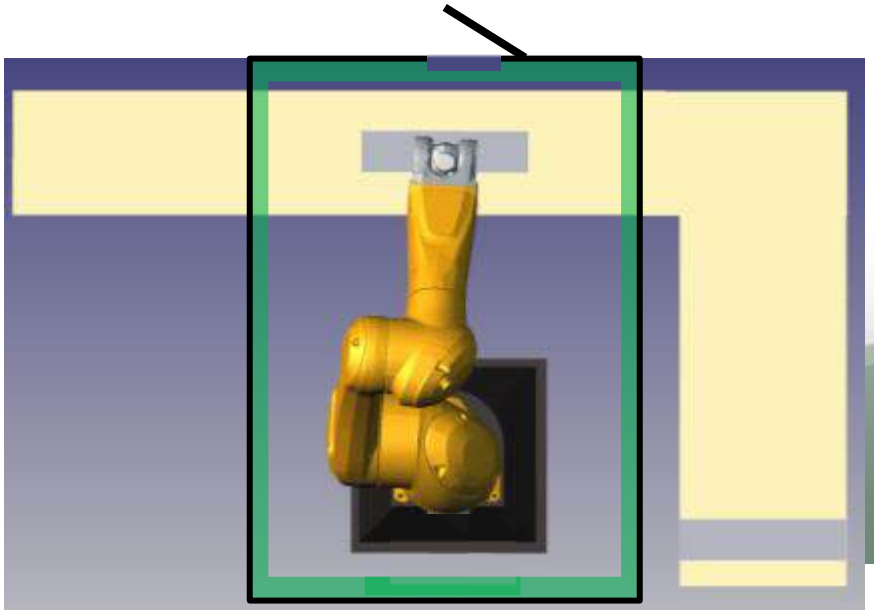
Słabsze wygrodzenia / kurtyny świetlne

Bezpieczne strefy ograniczające zasięg

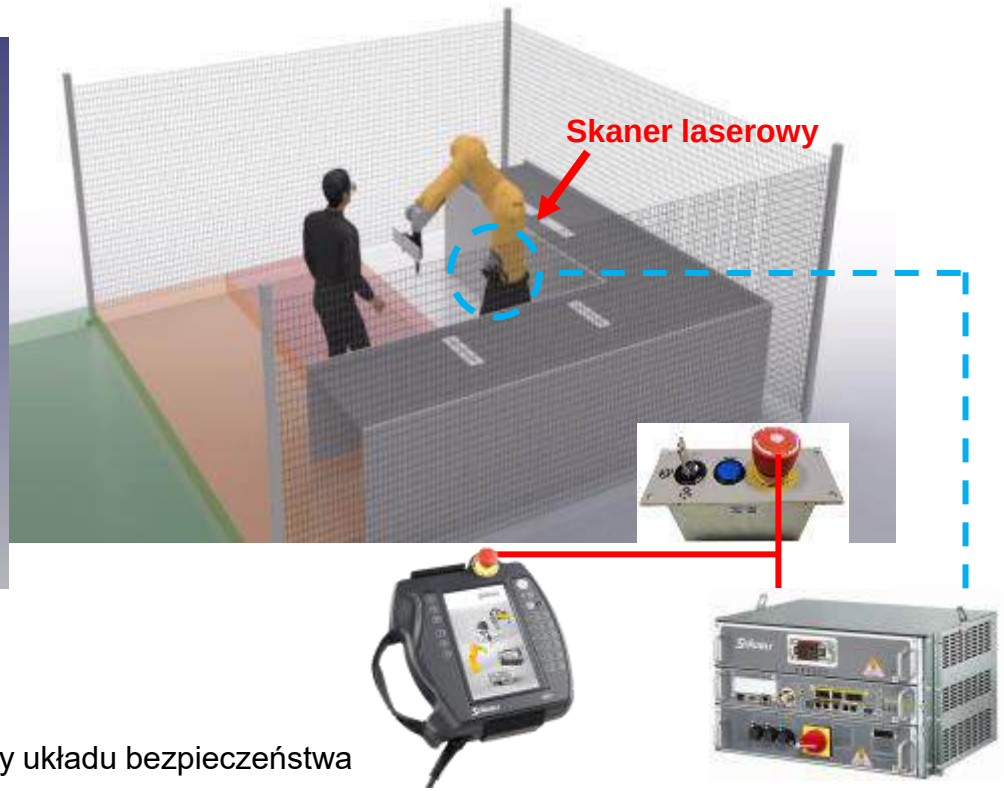
UKŁAD BEZPIECZEŃSTWA CS9

- W przypadku robotów kolaboracyjnych (współpracujących), układ bezpieczeństwa zakłada redukcję lub eliminację wygrodzeń

„Lekkie” wygrodzenie z drzwiami



„Lekkie” wygrodzenie, bez drzwi, przestrzeń współdzielona



„Lekkie” wygrodzenie lub kurtyna świetlna

Wydzielona przestrzeń

Stałe elementy układu bezpieczeństwa

Układ bezpieczeństwa aktywowany w trybie automatycznym (kurtyna świetlna, skaner laserowy, …)

STOPNIE WSPÓŁPRACY CZŁOWIEK – ROBOT (MRC)

Stopień 1 MRC: Mocne wygrodzenia oddzielające robota i człowieka. Praca wykonywana przez robota

Stopień 2 MRC: Zastosowanie wygrodzeń z kurtynami świetlnymi. Praca wykonywana przez robota. Operator okazjonalnie wchodzi w obszar pracy (wymian palety, załadunek podajnika wibracyjnego, ...).

Stopień 3 MRC: Zastosowanie kurtyn i skanerów laserowych. Robot i operator biorą udział w procesie produkcyjnym. Operator regularnie wchodzi w obszar pracy robota (Kontrola Jakości, przygotowanie detali do załadunku, …).

Stopień 4 MRC: Brak wygrodzeń. Robot i operator biorą udział w procesie produkcyjnym. Robot zatrzymuje się w momencie kontaktu z operatorem (Kontrola Jakości, przygotowanie detali do załadunku, ..).

Stopień 5 MRC: Brak wygrodzeń. Robot i operator biorą udział w procesie produkcyjnym. Robot i operator poruszają się jednocześnie.

Jeden robot do wszystkich stopni MRC

Stopień MRC

Kontakt niemożliwy

STOPIEŃ 1

Kontakt niepożądany

STOPIEŃ 2

Możliwy kontakt z zatrzymanym robotem

STOPIEŃ 3

Robot zatrzymuje się przed kontaktem

Kontakt niepożądany – zatrzymanie robota

STOPIEŃ 4

Kontakt dozwolony – praca jednoczesna

STOPIEŃ 5

Robot się nie zatrzymuje

Produktywność (prędkość robota)

Komplikacja analizy ryzyka

Przestrzeń pracy robota

Przestrzeń pracy operatora

Współdzielony obszar pracy

NA POCZĄTKU?

Po zatrzymaniu produkcji:

- Pierwszy cykl z małą prędkością (10 %)
  - zmiana prędkości za pomocą odpowiednich przycisków
- Bądź gotów zatrzymać robota:
  - **E-STOP** → natychmiastowe odcięcie zasilania silników i przerwanie ruchu (zatrzymuje całą linię) – używany w ostateczności
  - **wyłącznik zasilania ramienia** → zatrzymuje tylko robota
  - **przycisk MOVE/HOLD** → "łagodny" stop bez wyłączania serwonapędów

- Jeżeli cela jest do tego dostosowana (kolaboracja robot – operator) zatrzymanie może nastąpić po wyzwoleniu odpowiedniego czujnika np. laser czy kurtyna laserowa
- Zawsze przestrzegaj zasad bezpieczeństwa

TRYBY PRACY ROBOTA

- Automatyczny  LUB 

  LOKALNY ZDALNY

  - cela jest zamknięta, nie ma nikogo w środku
  - ramię jest kontrolowane przez program
  - ruchy ramienia mogą być wykonywane z dużą prędkością
  - tryb zdalny:  Robot jest kontrolowany przez PLC

- Ręczny (uczenie punktów, ruchy ręczne, dojazd do pozycji bazowej) 

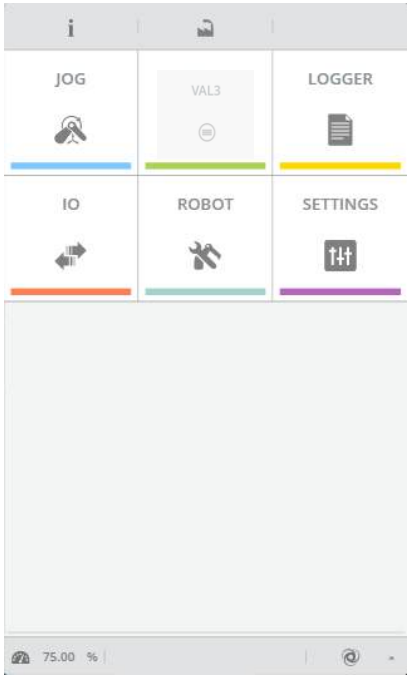
  MANUAL

  - robot jest kontrolowany przez operatora z Teachpendantem
  - program może być wykonywany przy użyciu przycisków «Move/Hold»
  - prędkość jest ograniczona do **250 mm/s**, operator może przebywać w pobliżu robota

URUCHOMIENIE ROBOTA

1 Załączenie głównego wyłącznika

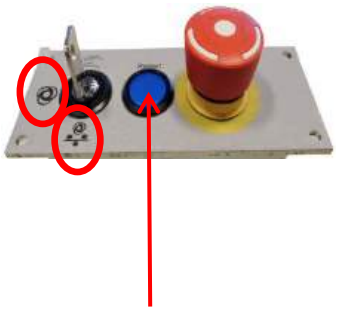
2 Poczekaj na wyświetlenie się głównego menu



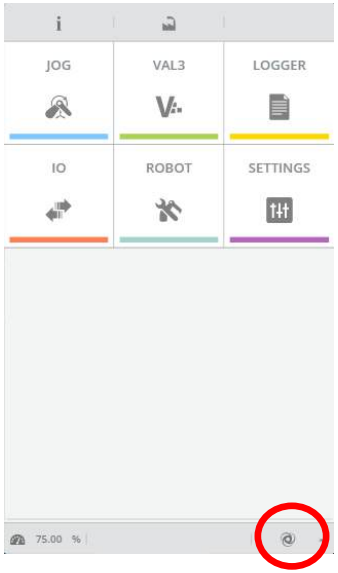
LUB

Aplikacja może być automatycznie załadowana i/lub uruchomiona, po czym może zostać wyświetlona strona interfejsu HMI

WŁĄCZANIE SERWONAPĘDÓW – tryb automatyczny



1 Brak E-Stop



3 Naciśnij « **Restart** » niebieski przycisk Safety!. Chyba, że wprowadzono zmiany w konfiguracji układu bezpieczeństwa.

2 Wybierz tryb automatyczny (lokalny lub zdalny)

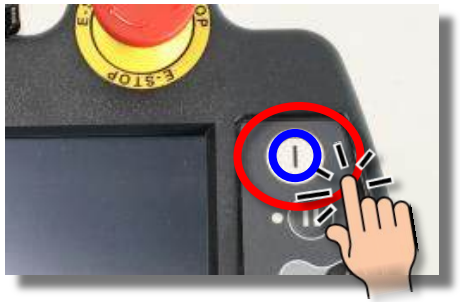
LOKALNY

LUB

ZDALNY

4 **LOKALNY**: Naciśnij przycisk załączania napędów

LOKALNY



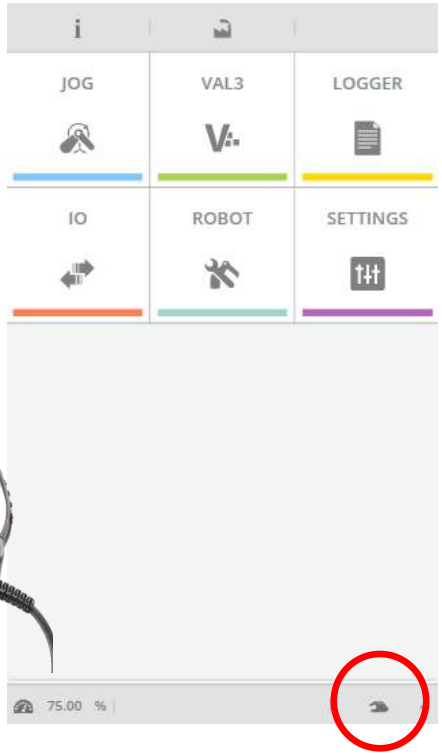
4 **ZDALNY**: Załączenie napędów realizowane przez urządzenie zewnętrzne np. PLC

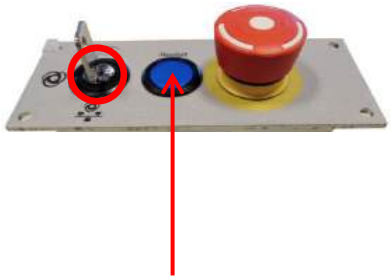
ZDALNY

Możliwość pracy bez WMS na trybie serwisowym – możliwa zmiana trybów z poziomu pilota

WŁĄCZANIE SERWONAPĘDÓW – tryb ręczny

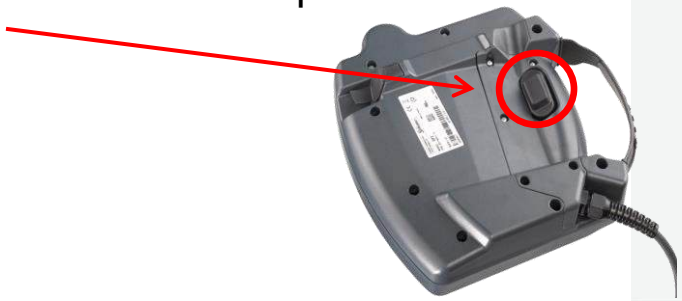
1 Wybierz tryb ręczny





2 Naciśnij « **Restart** » niebieski przycisk Safety! - tylko po uruchomieniu kontrolera

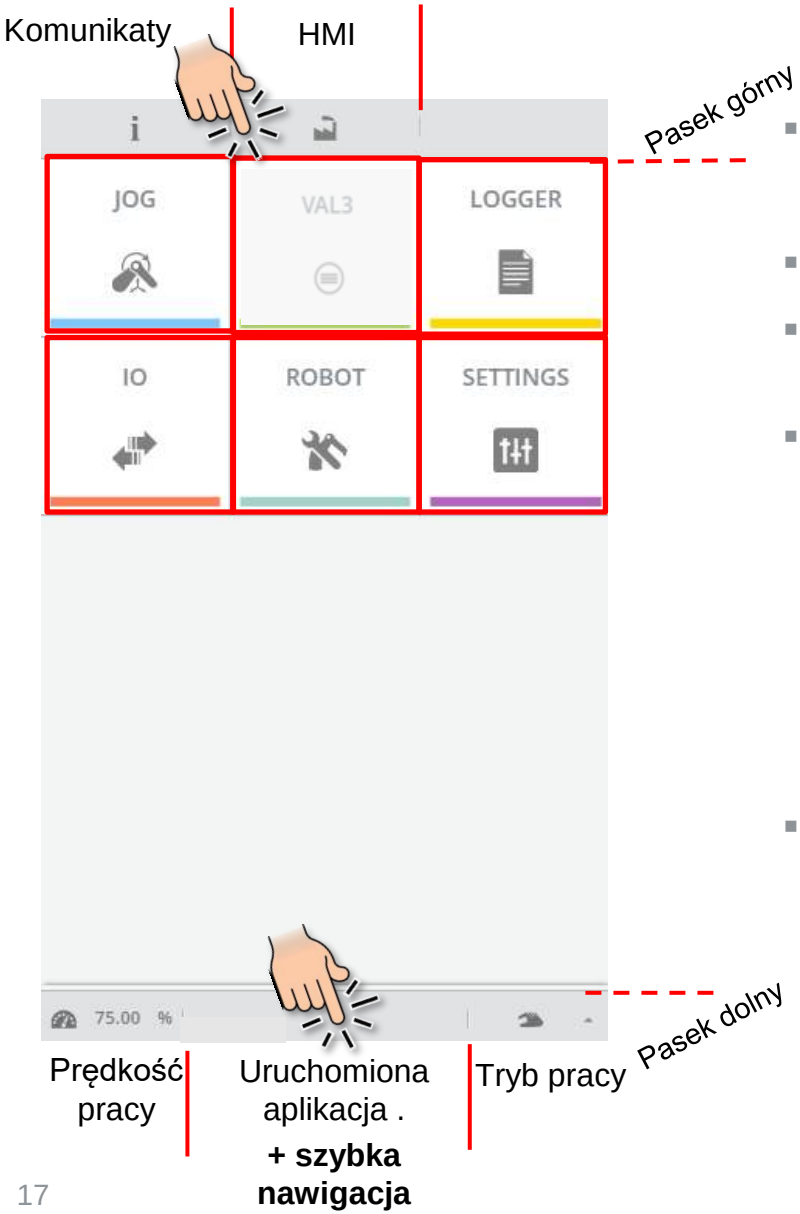
3 Wciśnij przycisk zezwolenia **Deadman** na Teachpendancie



4 Naciśnij przycisk załączania napędów

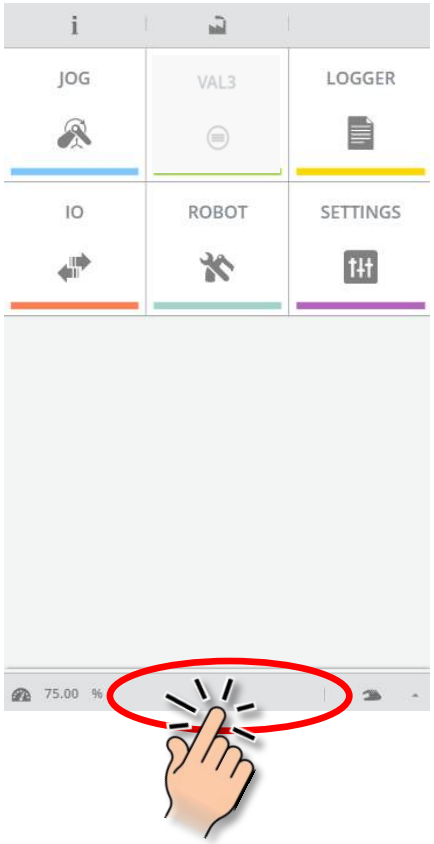
UWAGA: Napędy zostaną wyłączone po puszczeniu przycisku zezwolenia „**Deadman**”

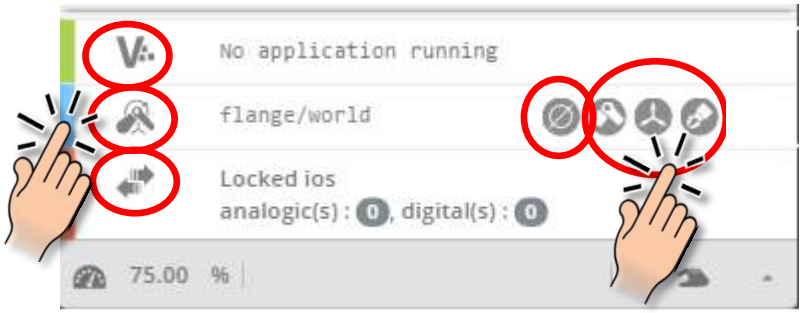
STRONA STARTOWA



- **JOG**: Ruch ręczny
  - nauka punktów, ruch do punktów
- **VAL3**: Aplikacja VAL3
  - otwieranie, uruchamianie, modyfikacja, zapis ...
- **LOGGER**: logi błędów
- **IO**: wejścia/wyjścia = komunikacja pomiędzy robotem i peryferiami (włączając sygnały układu bezpieczeństwa)
- **ROBOT**:
  - *calibration* (kalibracja), *recovery* (odzyskiwanie pozycji), *phasing*, *brake release* (zwalnianie hamulców),
  - informacje o wersjach sprzętu i oprogramowania
  - *recorder* (nagrywanie trajektorii), *safety* (układ bezpieczeństwa)
- **SETTINGS**: Ustawienia
  - *network* (adres IP), *language* (język menu), *date time* (data i godzina), *profiles (*zmiana profilu użytkownika)

SZYBKA NAWIGACJA

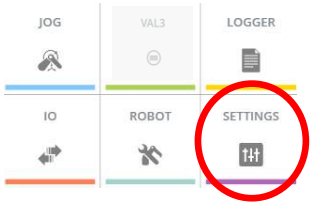




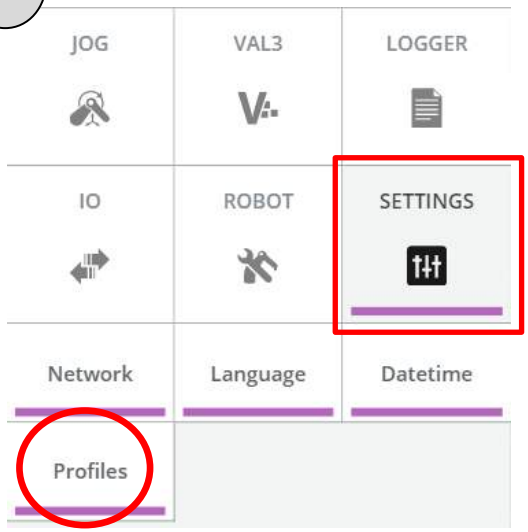
Szybki dostęp z każdego poziomu menu do:

- **Application manager** – kontrola aplikacji
- **Jog** – kontrola ruchu ramienia
- **I/O** – kontrola sygnałów I/O
- Szybki wybór trybu ruchu (Joint, Frame lub Tool) bez zmiany ekranu
- Rezygnacja z ręcznego ruchu ramienia ⊘ - uruchomienie aplikacji w trybie ręcznym

WYBÓR PROFILU UŻYTKOWNIKA

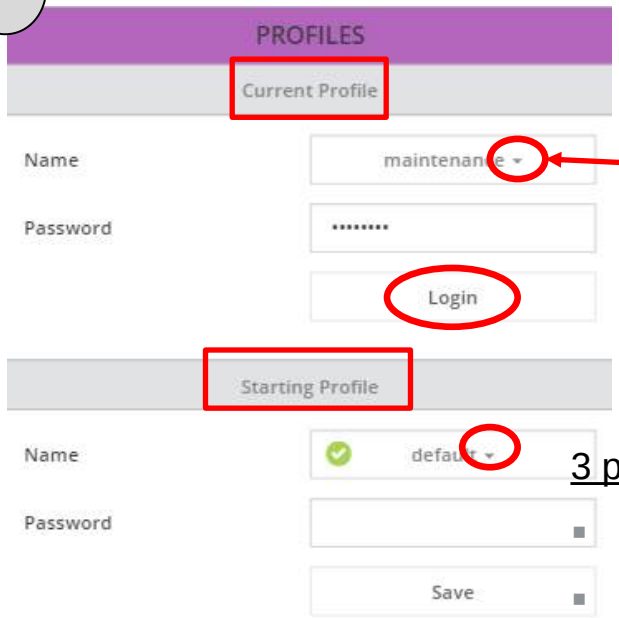


1



- 2 obszary: **Current profile** – aktualny profil
  Starting profile – profil startowy
- Niektóre profile są zabezpieczone hasłem
- Profil **Default** nie wymaga podania hasła

2



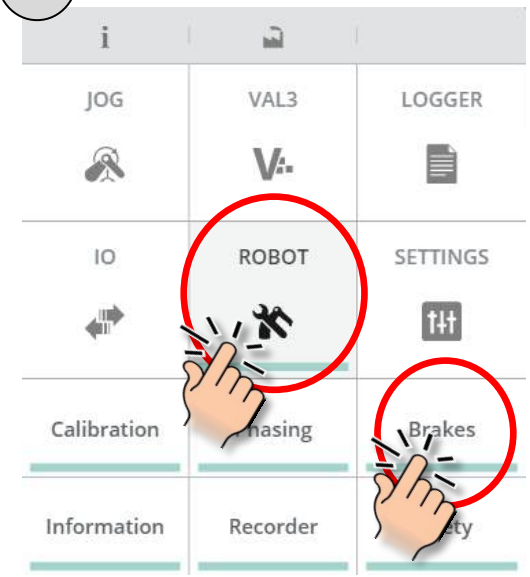
Lista dostępnych profili

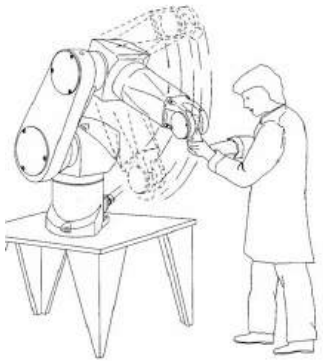
3 profile domyślne:

- **default** – bez hasła
- **maintenance** – chroniony hasłem
- **staubli** – chroniony hasłem

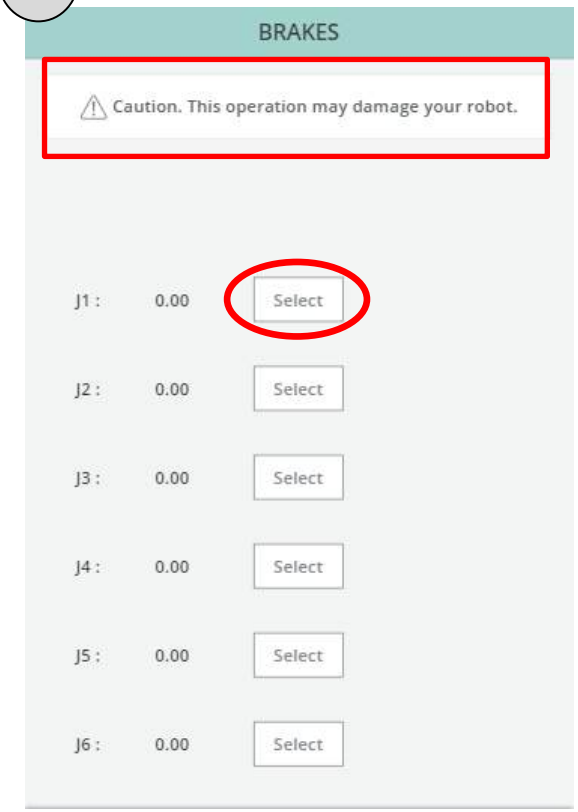
ZWALNIANIE HAMULCÓW

1

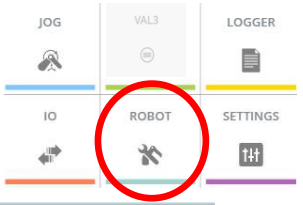


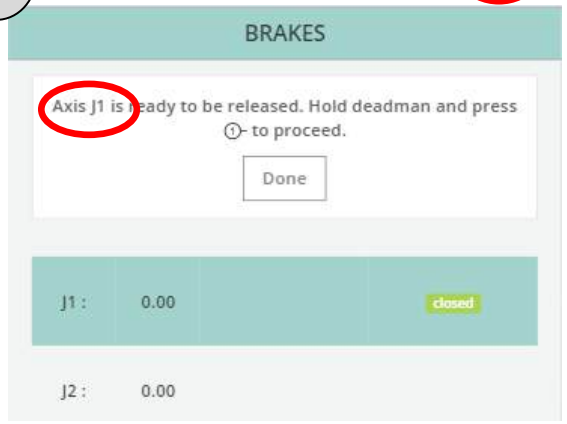


2

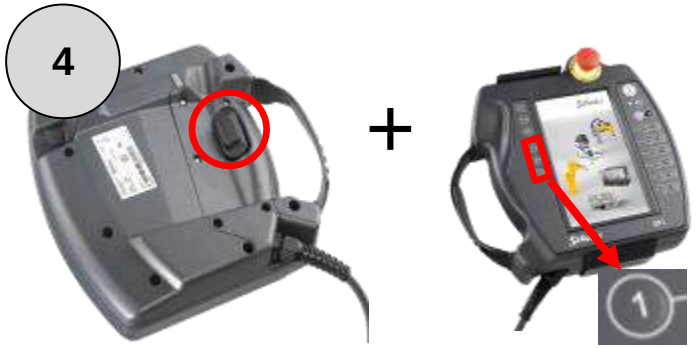


3





4



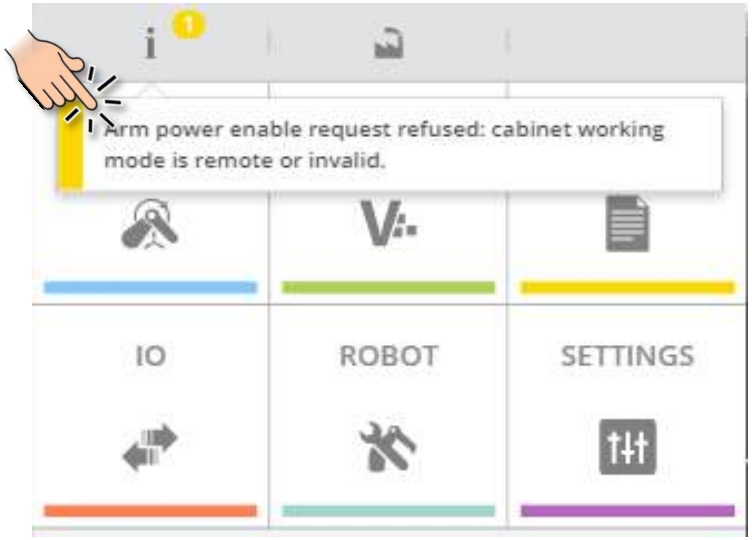
Alternatywnie: opcja R.B.R

- Zmniejszona prędkość opadania ramienia
- Może być używany gdy kontroler jest wyłączony – wymaga zewnętrznego zasilacza 24V

!!! Przytrzymaj ramię zanim zwolnisz hamulce !!!

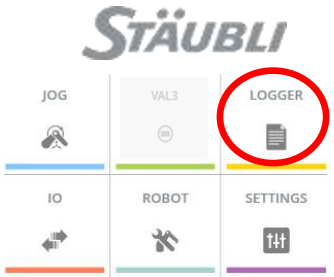
KOMUNIKATY

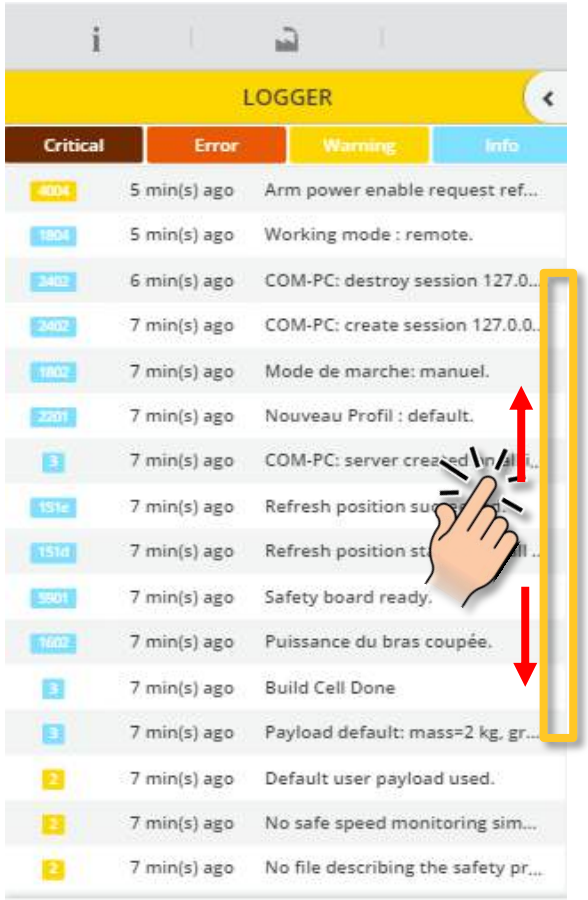
- Kontroler informuje użytkownika o statusie systemu za pomocą komunikatów wyświetlanych na ekranie i pozostających na nim przez kilka sekund
- Dotknięcie obszaru « **i** » otwiera logi błędów



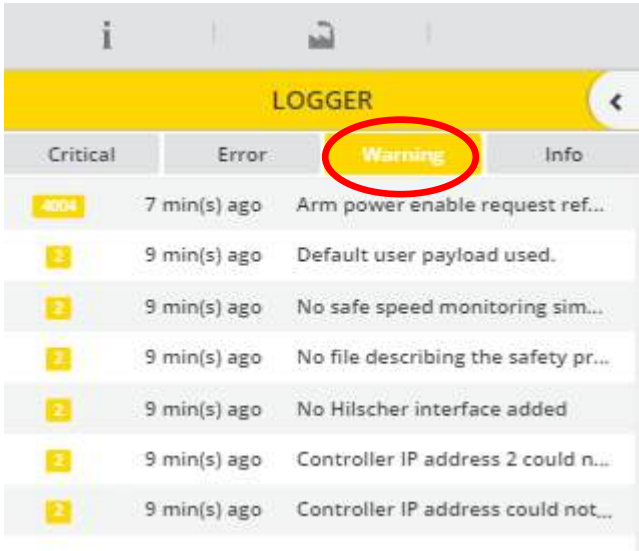
- Kolor pozwala na identyfikację rodzaju komunikatu:
  - **Critical** Błędy krytyczne – np. poważna usterka części elektronicznej
  - **Error** Błędy w programie itp.
  - **Warning** Ostrzeżenia dotyczące obsługi robota itp.
  - **Info** Np. zmiana trybu pracy, itp.

LOGI BŁĘDÓW





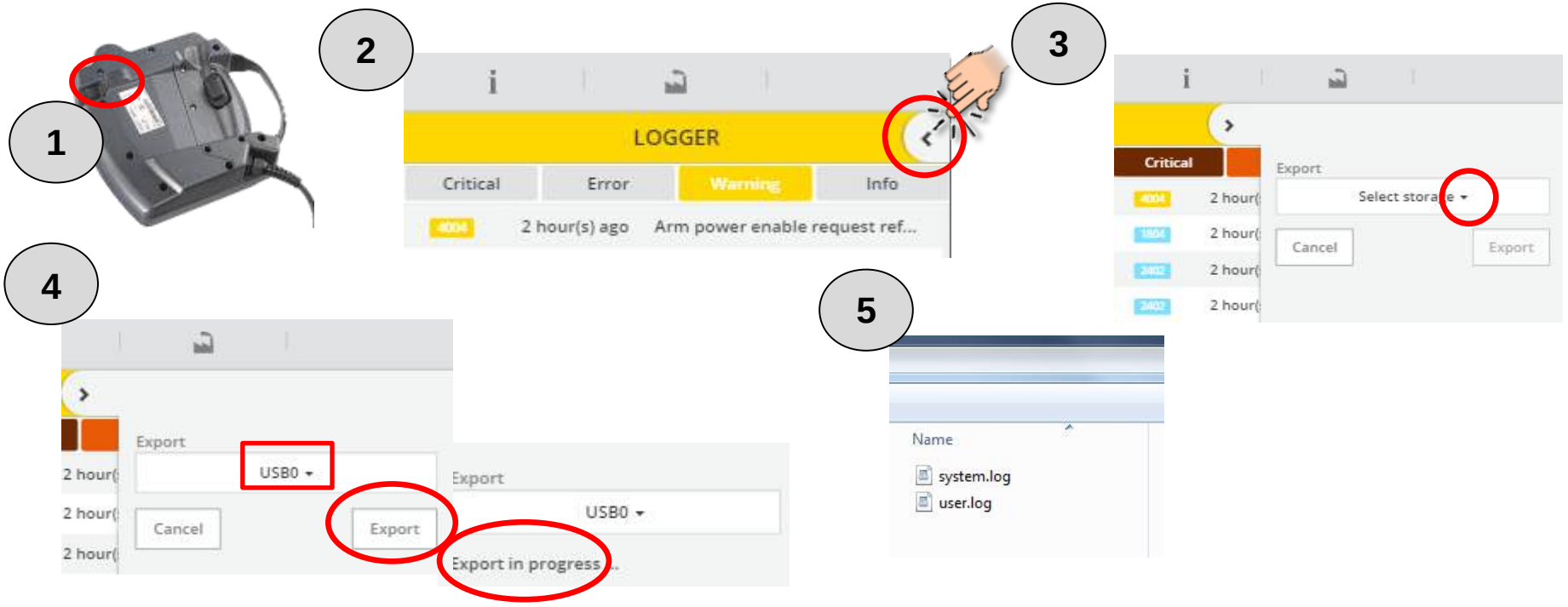
- Logi błędów są dostępne z poziomu głównego menu lub poprzez naciśnięcie obszaru « **i** »
- Logi są uszeregowane względem daty i godziny wystąpienia
- Możliwość zastosowania filtra wyświetlanych logów błędów: Critical, Error, Warning, Info
- Przykład – wyświetlanie logów ***Warning***



- Przewijanie listy za pomocą palca
- Wyświetlenie pełnego komunikatu po kliknięciu palcem na komunikacie

EKSPORT LOGÓW BŁĘDÓW

1. Podłącz pamięć USB do kontrolera lub teachpendanta (FAT lub FAT32)
2. Z poziomu menu logów błędów, wybierz menu boczne
3. Wybierz USB0 jako pamięć docelową
4. Naciśnij przycisk Export
5. Pliki mogą być odczytane za pomocą dowolnego edytora tekstu lub SRS (zalecane)



NAWIGACJA PO MENU – ćwiczenie

Ćwiczenie nr 1: Nawigacja po menu robota

- Proszę uruchomić kontroler
- Nawigacja po menu – zapoznać się z menu
- Sprawdzić profil użytkownika
- Załączyć serwonapędy w różnych trybach pracy (ręczny, automatyczny lokalny)
- Zwolnić hamulce na poszczególnej osi (proszę o uwagę i rozsądek)
- Proszę odczytać logi oraz zgrać je na pendrive'a

CZĘŚĆ 1 – OPERATOR CS9

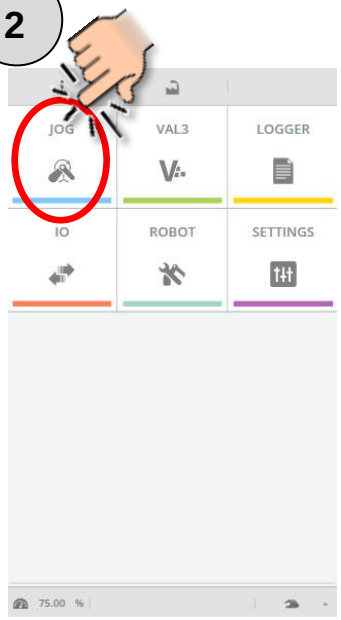
TRYBY RUCHU RAMIENIA

PORUSZANIE RAMIENIEM ZA POMOCĄ SP2

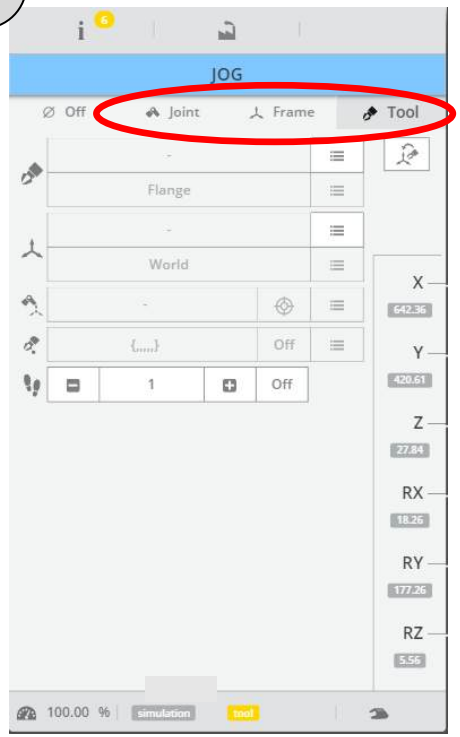
1 Wybierz ręczny tryb pracy

2 sposoby wyboru trybu ruchu

2



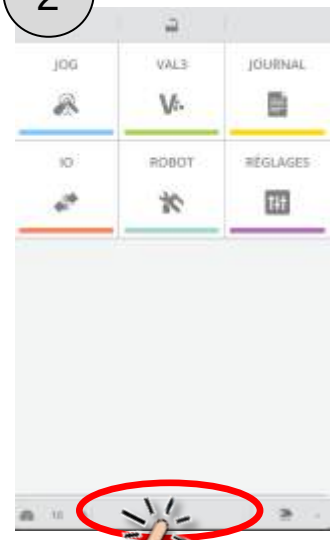
3



Klasyczny:

- Dostęp do: punktów, narzędzi, układów współrzędnych
- Dostęp do rodzajów ruchu: joint, frame, tool, appro itp.

2



3

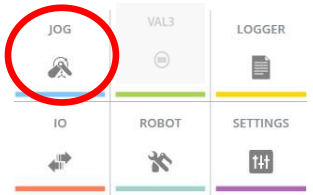
Szybki dostęp z dowolnego miejsca:

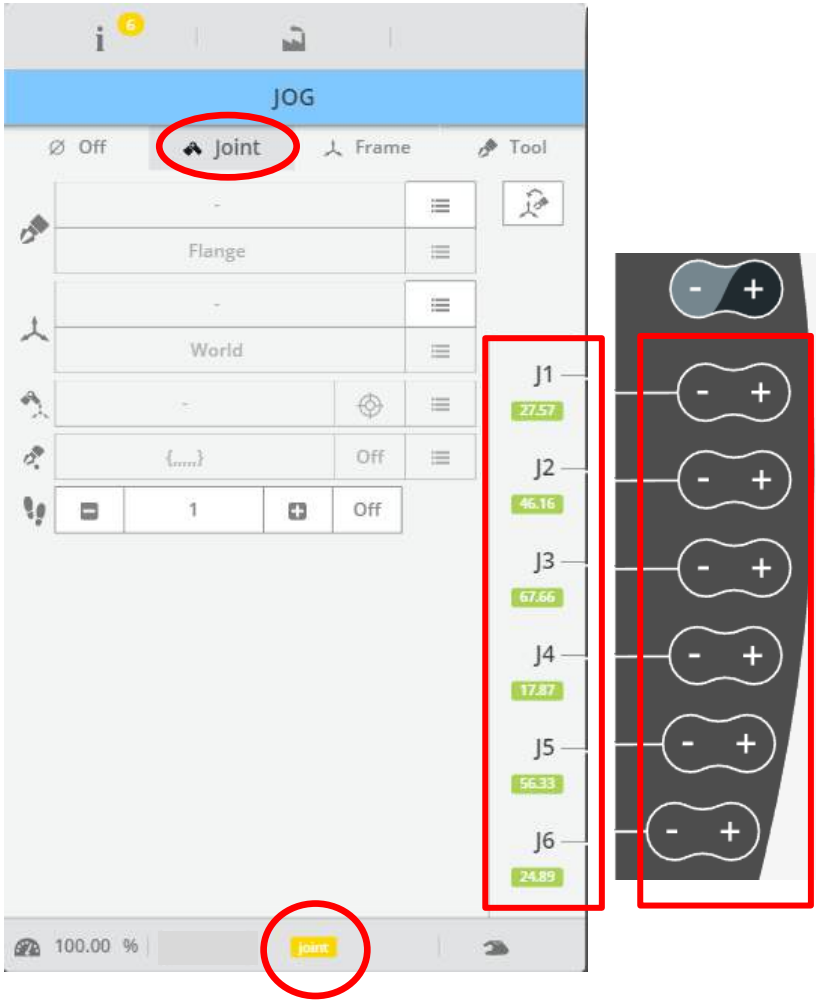
Środek dolnego paska menu

- Szybki wybór rodzaju ruchu bez zmiany menu

PORUSZANIE ROBOTEM: JOINT – tryb ręczny

- Menu **JOG** jest dostępne z poziomu głównego. Wybór trybu **JOINT** znajduje się poniżej niebieskiego paska JOG (obowiązkowo tryb ręczny!)

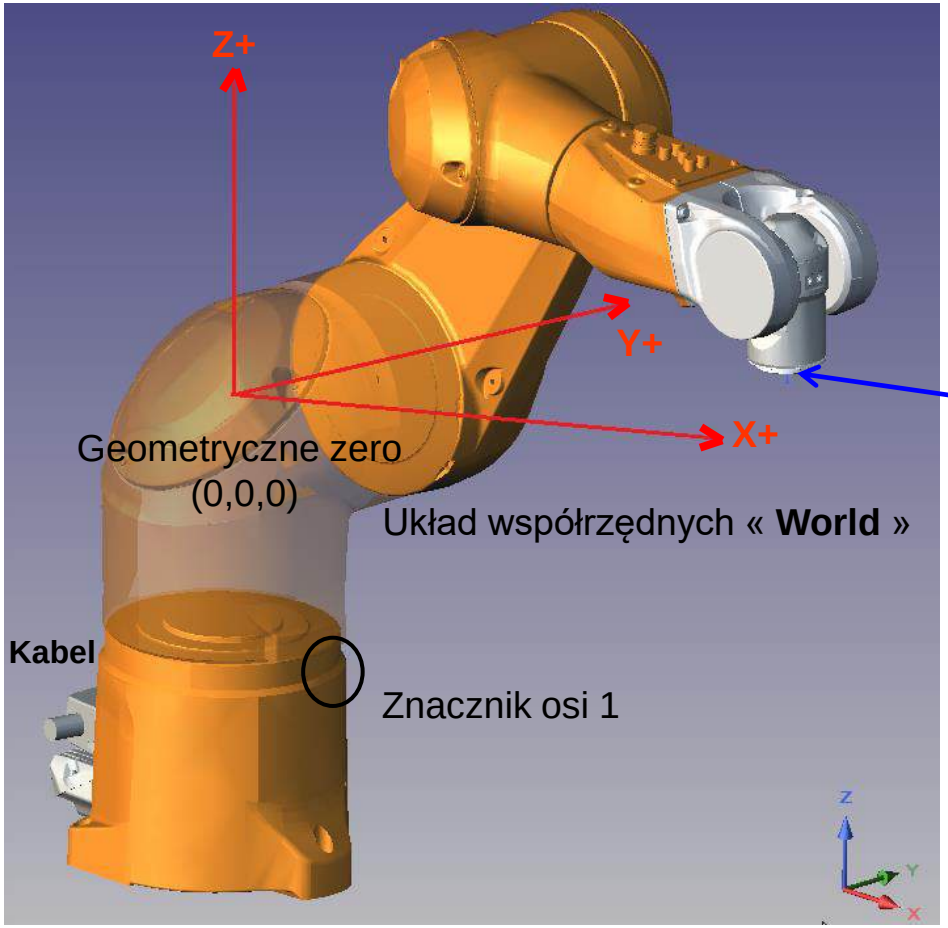




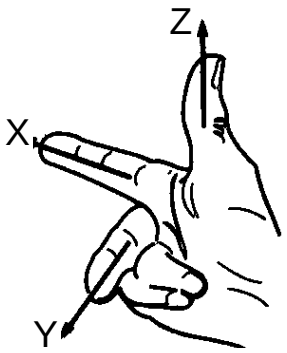
- Wybierz tryb **Joint**
- Aktualna pozycja robota jest wyświetlana w prawej części ekranu, obok przycisków ruchu osiami. Wyświetlane są pozycje kątowe wszystkich osi.
- Naciśnięcie przycisków + / - powoduje ruch osiami robota
- Informacja o aktywacji trybu **Joint** jest wyświetlana na dolnym pasku menu

PORUSZANIE ROBOTEM: FRAME – tryb ręczny

« **World** » jest domyślnym układem kartezjańskim, zaczepionym w podstawie robota



- Współrzędne {X,Y,Z}: wyliczane są od początku układu World do TCP (Tool Center Point)
- Domyślny punkt TCP znajduje się w środku flanszy osi 6

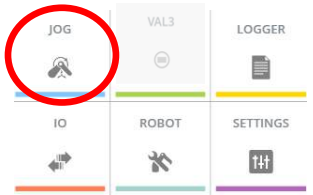


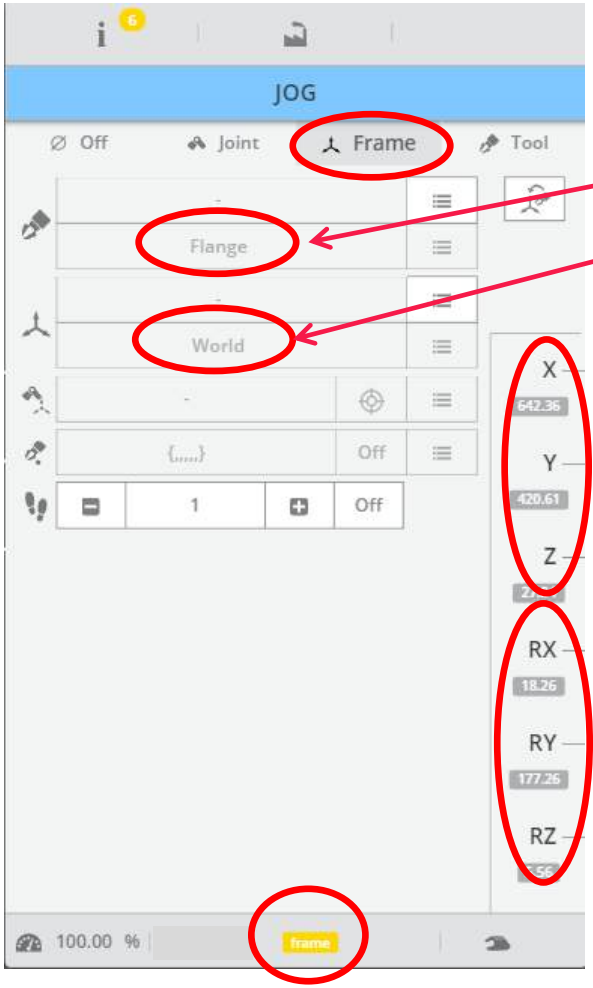
Prawa dłoń

Układ współrzędnych **World** jest stały, kierunki osi X, Y, Z są zawsze takie same

PORUSZANIE ROBOTEM: FRAME – tryb ręczny c.d.

- Menu **JOG** jest dostępne z poziomu głównego. Wybór trybu **FRAME** znajduje się poniżej niebieskiego paska JOG (obowiązkowo tryb ręczny!)
- Informacja o aktywacji trybu **Frame** jest wyświetlana na dolnym pasku menu





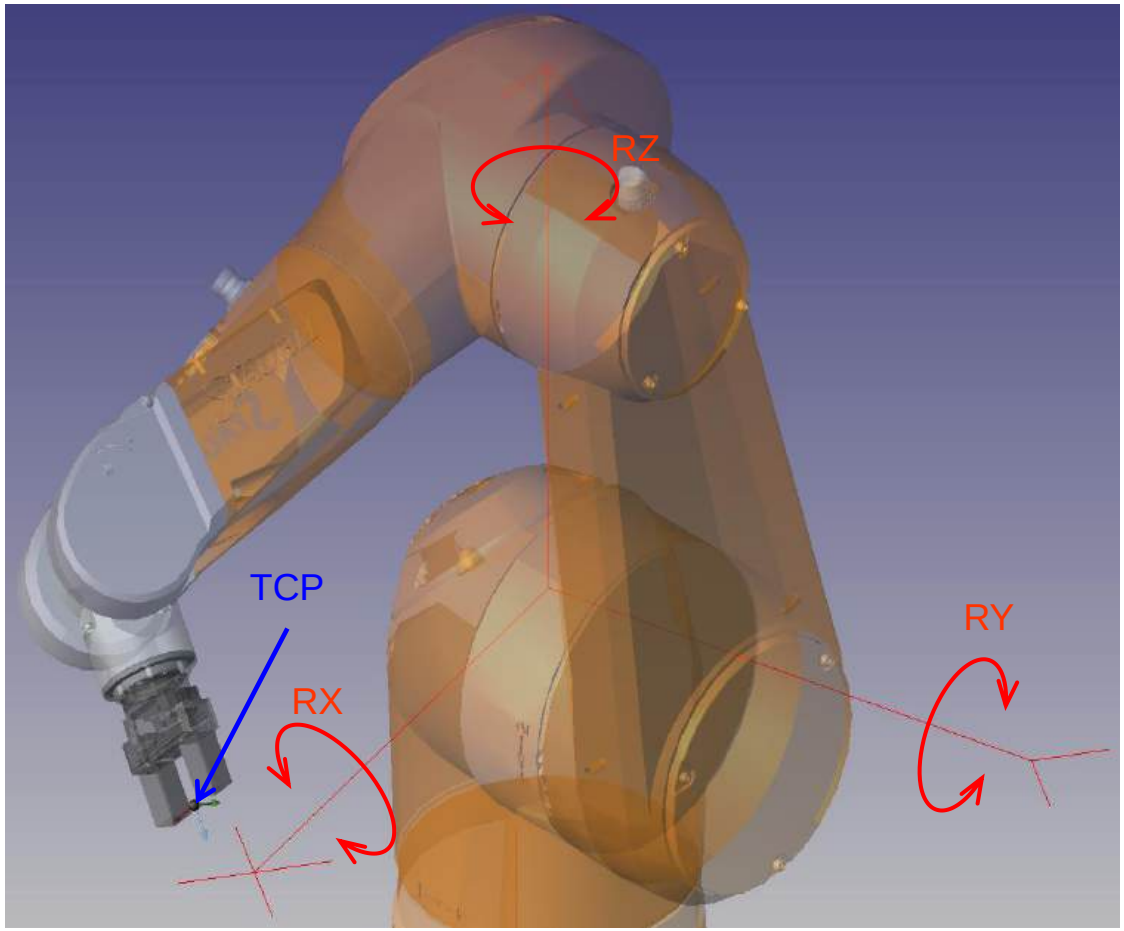
- Wybrane narzędzie z danej aplikacji
- Wybrany układ współrzędnych z danej aplikacji

- **X-Y-Z** współrzędne punktu TCP aktualnie wybranego narzędzia w danym układzie współrzędnych w mm

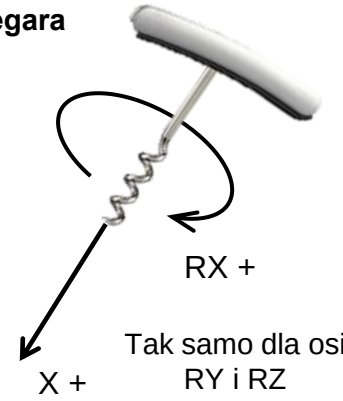
- **RX-RY-RZ** kąty nachylenia osi aktualnie wybranego narzędzia względem osi układu współrzędnych w stopniach

Można wybrać dowolny układ współrzędnych ze wszystkich aplikacji

ORIENTACJA NARZĘDZIA W UKŁADZIE WSPÓŁRZĘDNYCH

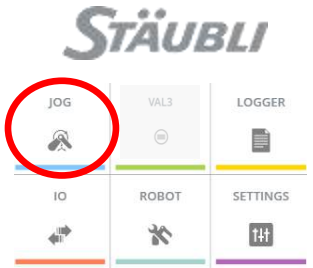


Obrót zgodnie z ruchem wskazówek zegara

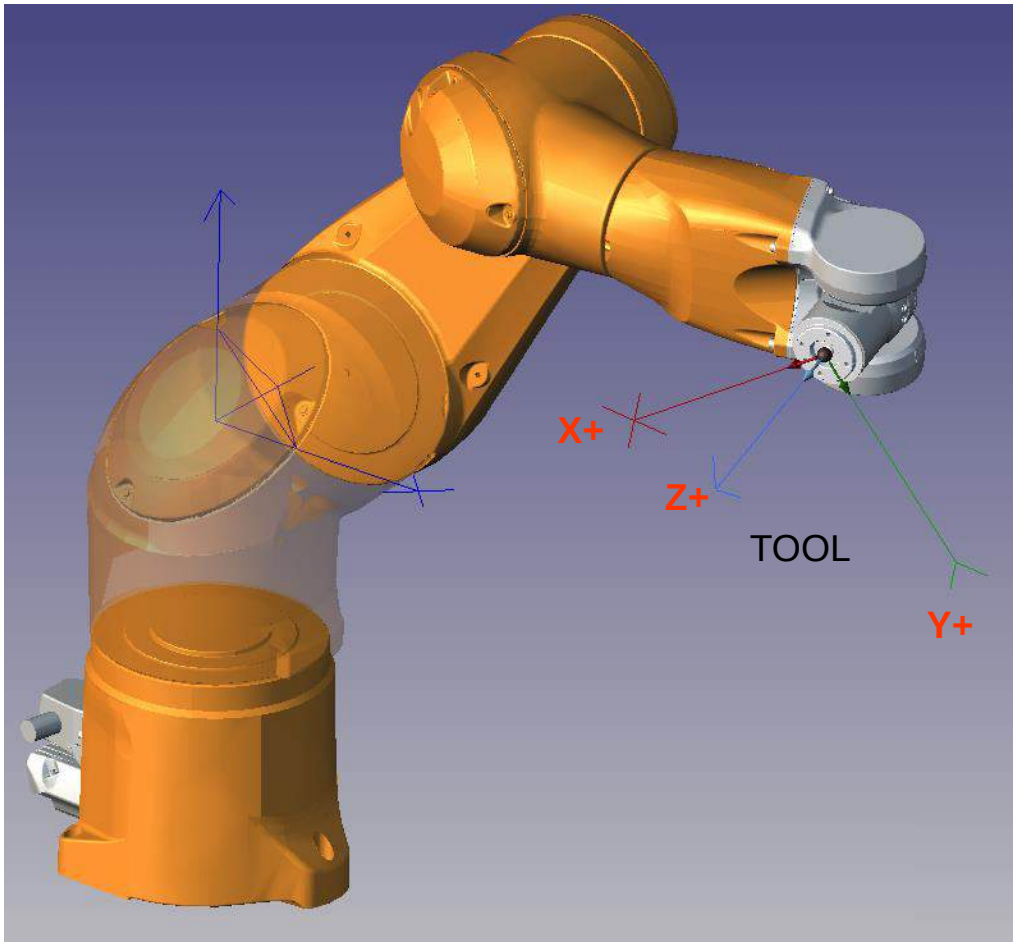


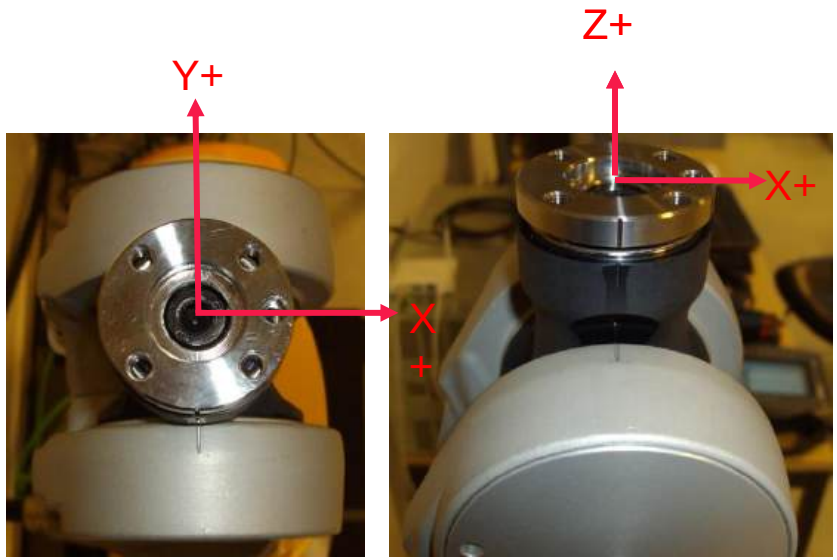
{RX,RY,RZ}:
Współrzędne orientacji punktu TCP narzędzia
=
Kąty nachylenia punktu TCP względem osi danego układu współrzędnych

PORUSZANIE ROBOTEM: TOOL – tryb ręczny



Poruszanie punktem TCP względem osi narzędzia



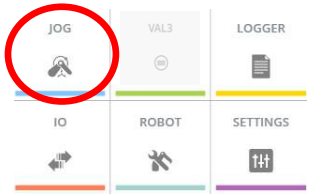


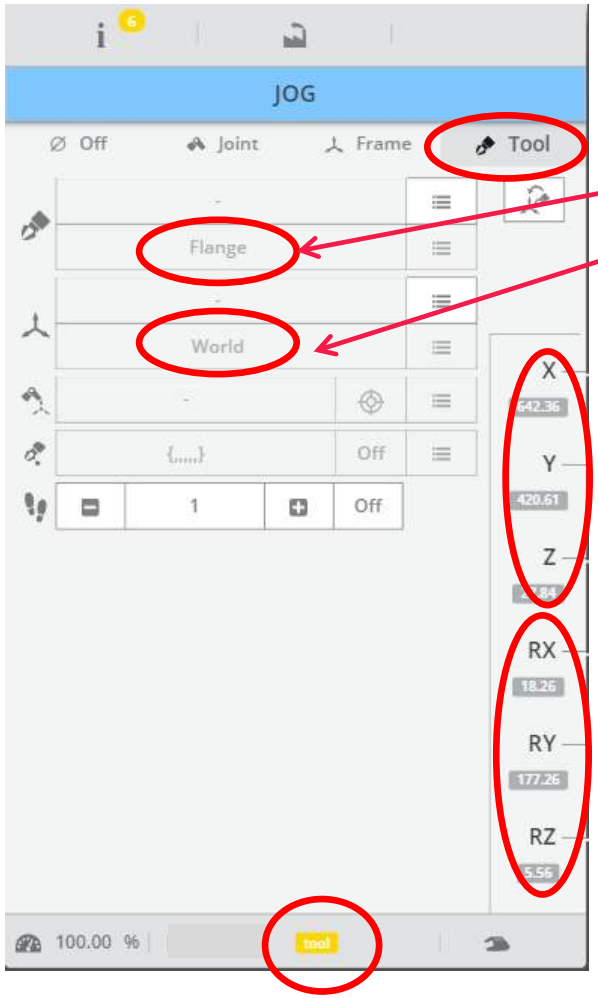
Domyślnym narzędziem jest « **Flange** », na osi 6 robota

UKŁAD RUCHOMY:
- Kierunki osi **X-Y** zmieniają się wraz z obrotem osi 6
- Kierunek osi **Z** „wychodzi” z flanszy robota

PORUSZANIE ROBOTEM: TOOL – tryb ręczny c.d.

- Menu **JOG** jest dostępne z poziomu głównego. Wybór trybu **TOOL** znajduje się poniżej niebieskiego paska JOG (obowiązkowo tryb ręczny!)
- Informacja o aktywacji trybu **Tool** jest wyświetlana na dolnym pasku menu





- Wybrane narzędzie z danej aplikacji
- Wybrany układ współrzędnych z danej aplikacji

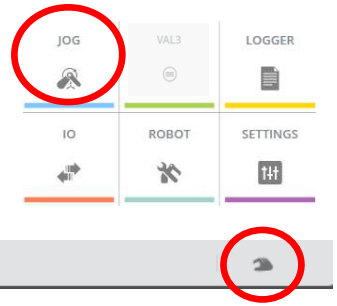
- **X-Y-Z** współrzędne punktu TCP aktualnie wybranego narzędzie w danym układzie współrzędnych w mm

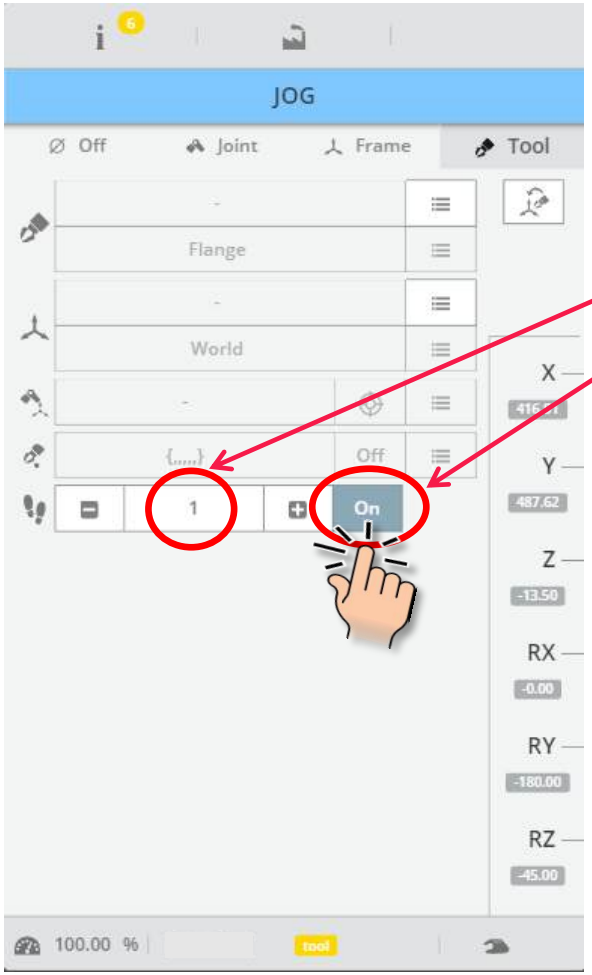
- **RX-RY-RZ** kąty nachylenia osi aktualnie wybranego narzędzia względem osi układu współrzędnych w stopniach

Można wybrać dowolny układ współrzędnych ze wszystkich aplikacji

PORUSZANIE ROBOTEM: RUCH KROKOWY

- Ruch krokowy jest możliwy dla wszystkich trybów ruchu (Joint, Frame lub Tool), w menu JOG
- Należy sprawdzić parametry ruchu krokowego (włączanie, wartość kroku) przed wykonaniem jakichkolwiek ruchów



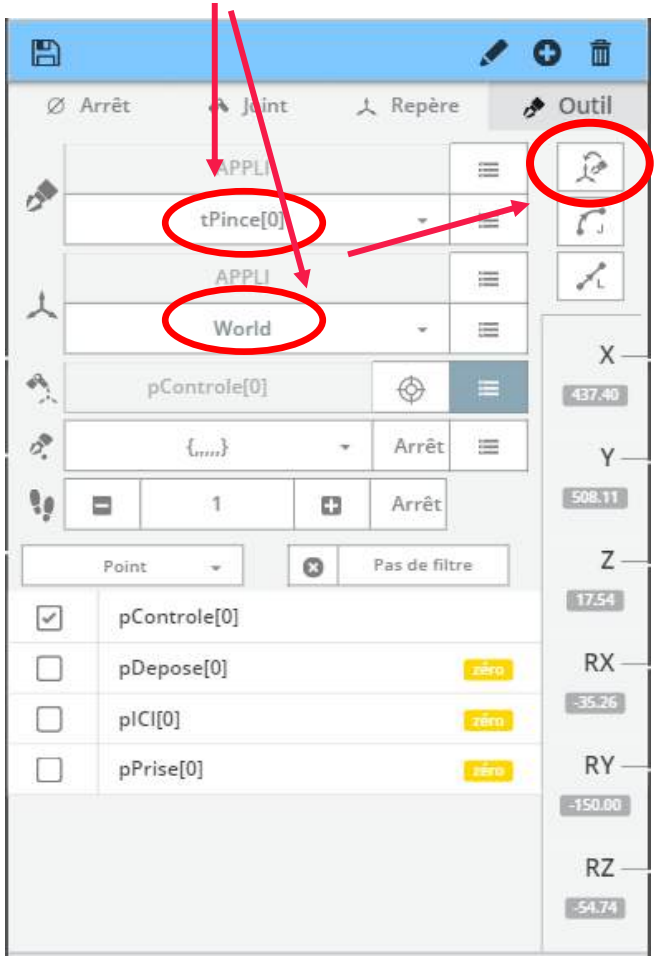


- Wprowadź wartość kroku
- Włącz tryb krokowy
- Przykład: 10 mm względem osi X, Y, Z lub 10 ° obrotu RX, RY lub RZ
- Pamiętaj o wyłączeniu trybu krokowego aby powrócić do trybu JOG

TRYB RUCHU POINT – USTAWIENIE NARZĘDZIA PROSTOPADLE DO OSI UKŁADU WSPÓŁRZĘDNYCH

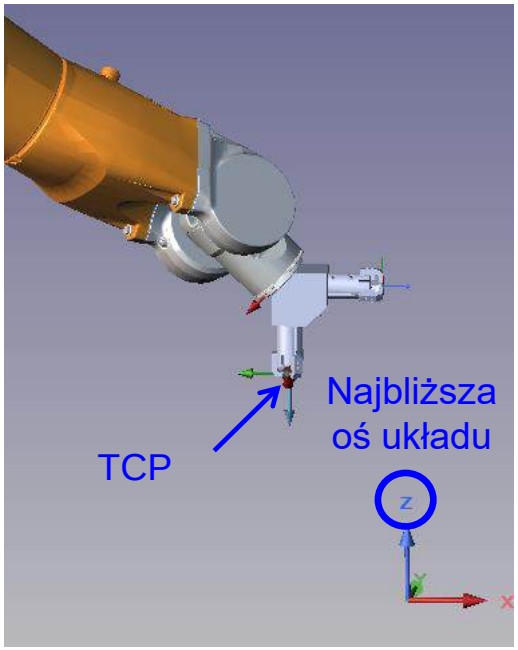
1

Przycisk  (*align*) pozwala na ustawienie osi Z **aktualnie** wybranego narzędzia równolegle do najbliższej osi X, Y lub Z **aktualnie** wybranego układu współrzędnych



2

Naciśnij i przytrzymaj przycisk Move/Hold, aby ustawić narzędzie



Po osiągnięciu pozycji, w dolnym pasku menu pojawi się informacja <REACHED>

PORUSZANIE RAMIENIEM ROBOTA – ćwiczenie

Ćwiczenie nr 2: Budowanie wieży

Proszę zbudować jak najwyższą wieżę z dostępnych pinów

Proszę wykorzystać:

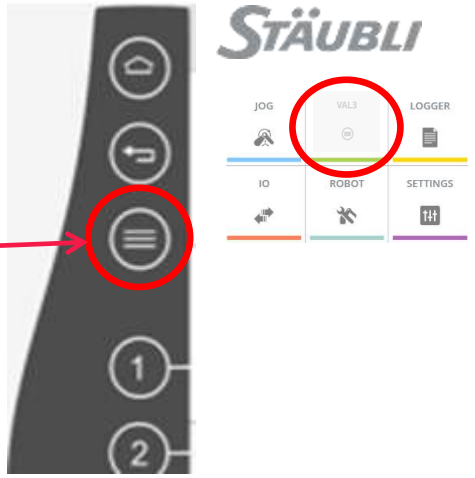
- Ruch **Joint**
- Ruch **Frame**
- Ruch **Tool**
- Ustawienie robota **Align**
- Ruch krokowy

CZĘŚĆ 1 – Operator CS9

APPLICATION MANAGER

APPLICATION MANAGER

Menu zarządzania aplikacjami napisanymi w języku VAL, jest wyświetlane na stronie głównej, jednak jest niedostępne. Dostęp do menu aplikacji, można uzyskać po naciśnięciu przycisku dodatkowego menu

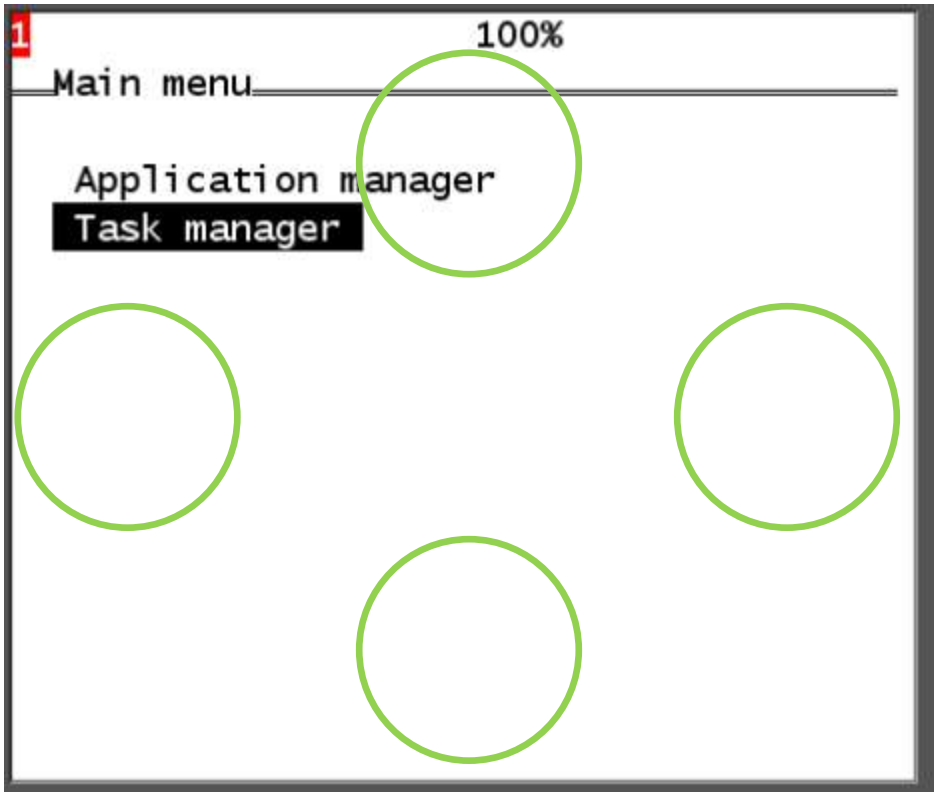


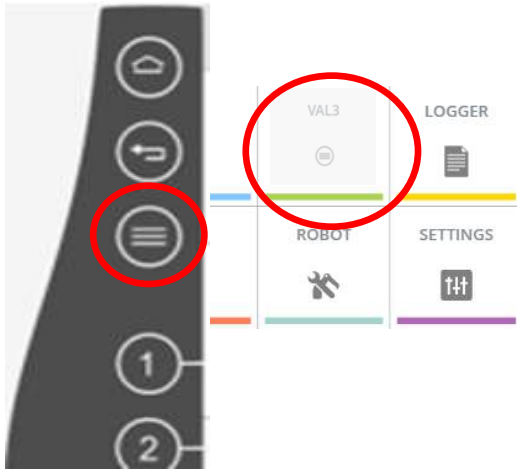
Menu zarządzania aplikacjami składa się z 2 pozycji:

- **Application Manager** – zarządzanie, otwieranie, zamykanie i edycja aplikacji
- **Task Manager** – debugging i uruchamianie krok po kroku
- Przyciski **RUN** oraz **STOP** służą do uruchamiania i zatrzymywania aplikacji
- Nawigacja po menu odbywa się za pomocą klawiatury ekranowej

APPLICATION MANAGER – NAWIGACJA PO MENU

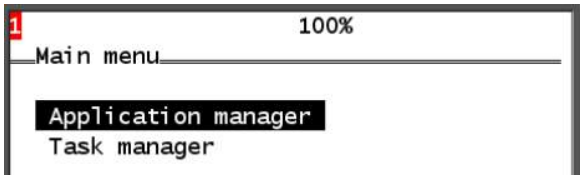
- Alternatywą dla przycisków strzałek, jest klikanie w górny, dolny, lewy lub prawy obszar ekranu. Zielony okrąg wskazuje miejsce kliknięcia

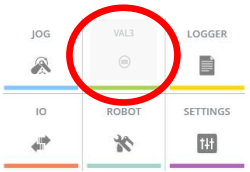




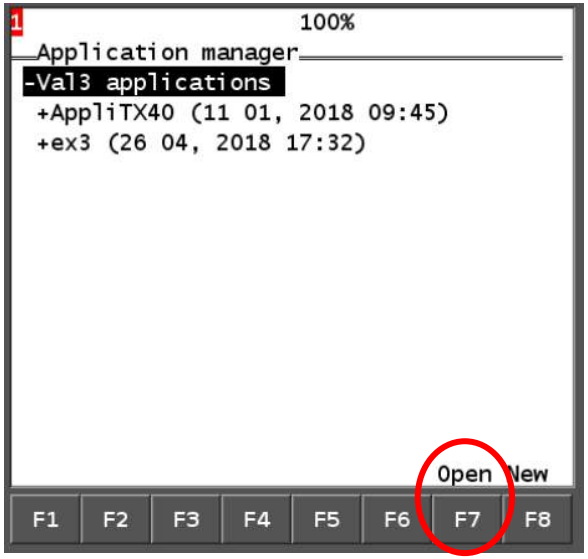
RĘCZNE OTWIERANIE APLIKACJI

1 Przycisk dodatkowego menu



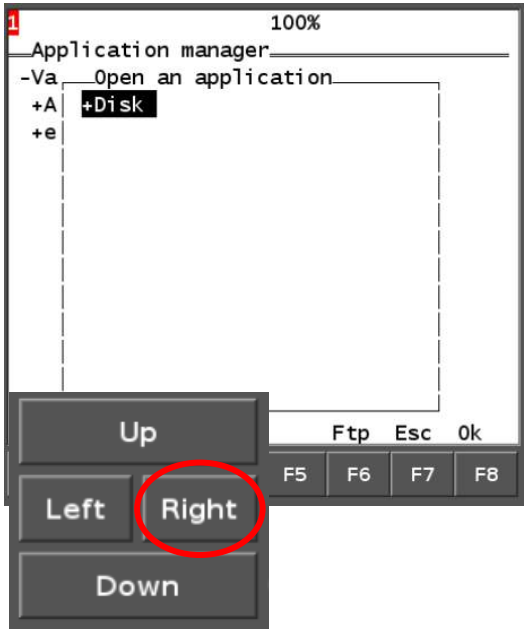


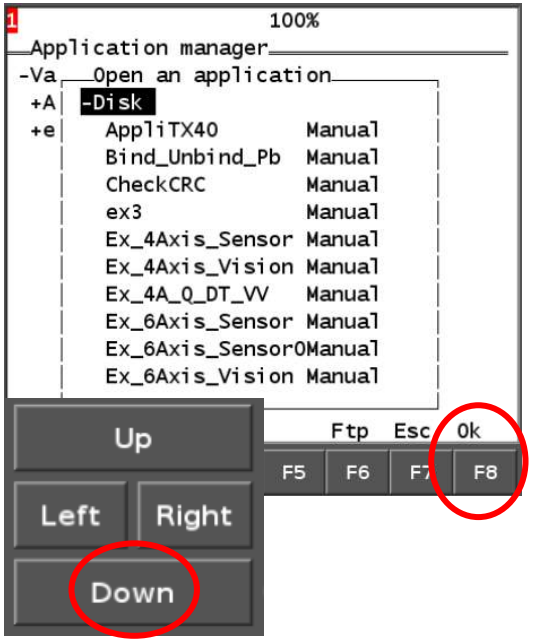
2 Wejście do Application manager (ENTER lub przycisk prawej strzałki lub dotknięcie prawej strony ekranu)



3 Przycisk F7 (Open)

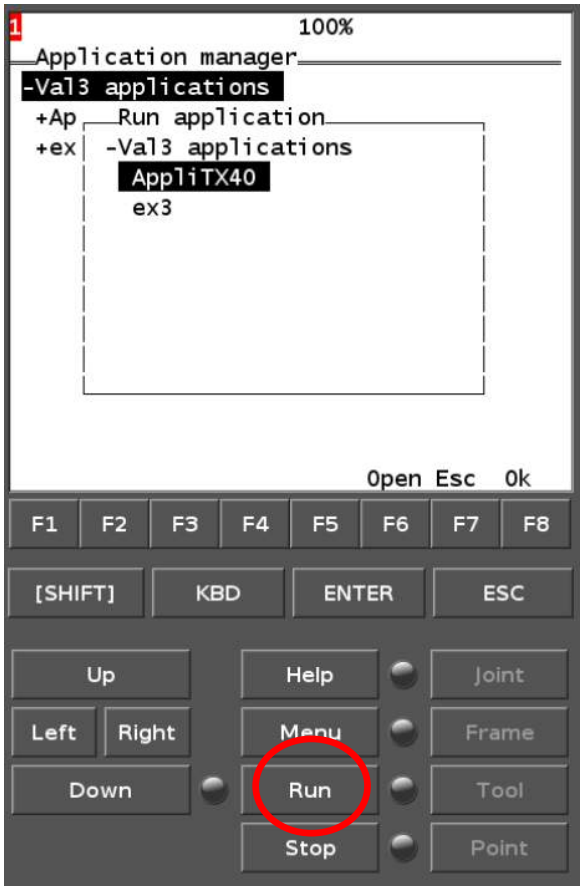
4
- Przycisk prawej strzałki lub dotknięcie prawej strony ekranu rozwija menu +Disk (lub pamięci USB jeżeli jest podłączona)
- Wybierz aplikację za pomocą kursora
- Zatwierdź przyciskiem F8 (OK) lub ENTER





RĘCZNE URUCHAMIANIE APLIKACJI

1 Przycisk RUN

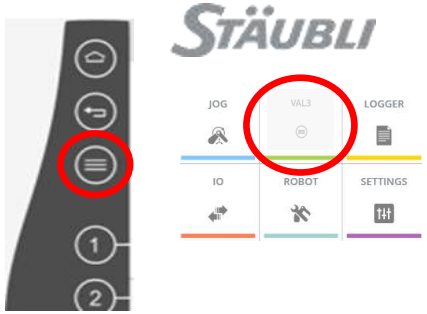


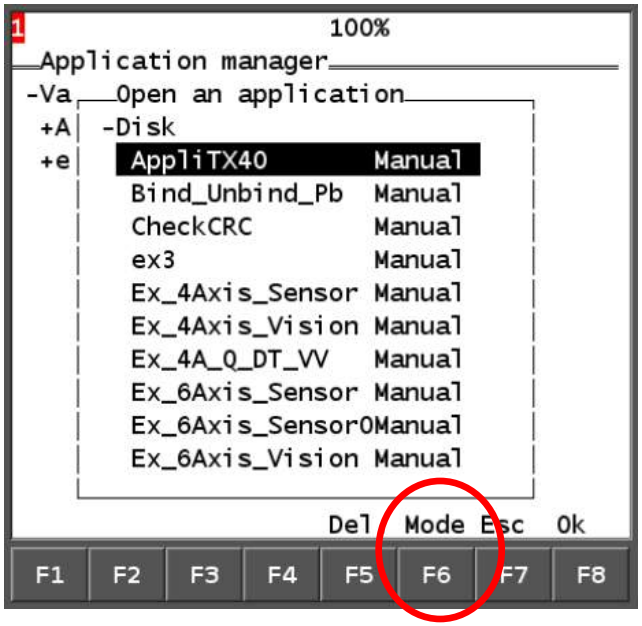
2

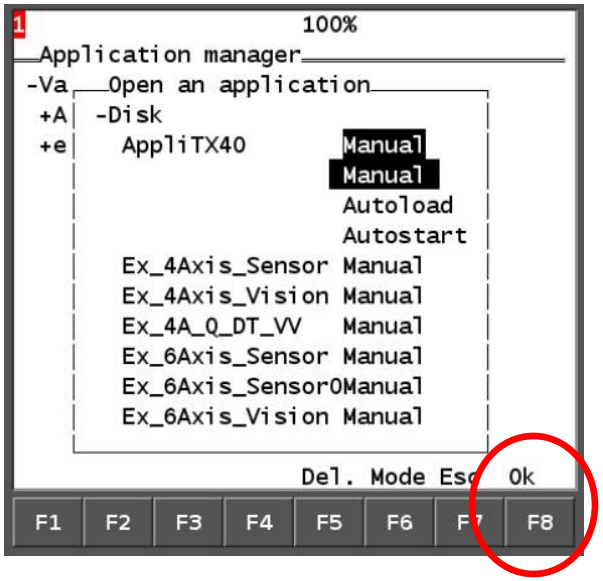
Wybór aplikacji z listy i zatwierdzenie przyciskiem **F8** (**Ok**) lub **ENTER**

Uruchomienie aplikacji jest sygnalizowane zapaleniem się diody sygnalizacyjnej obok przycisku **RUN**

TRYBY OTWIERANIA APLIKACJI







STRUKTURA APLIKACJI

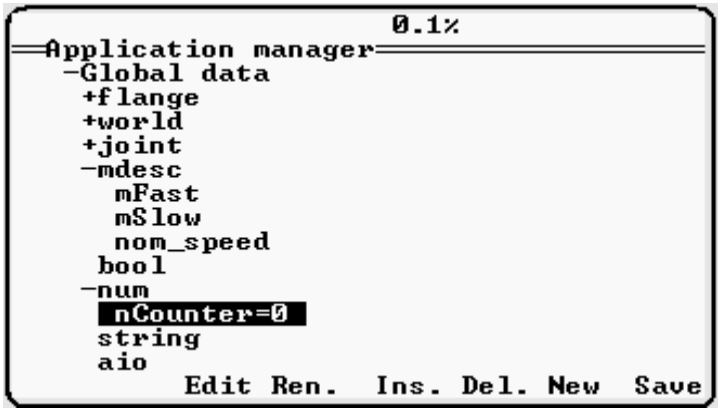
- **ZMIENNE GLOBALNE**: zmienne różnego typu, oraz zdefiniowane przez programistę:
  - **tool** – narzędzia, **local frame** – układy współrzędnych, **punkty Kartezjańskie** i **Joint**,
  - zmienne **Boolean**, **numeryczne**, **string** – ciągi znaków

- **PROGRAMY**: wszystkie programy zawarte w aplikacji

START() 

STOP() 

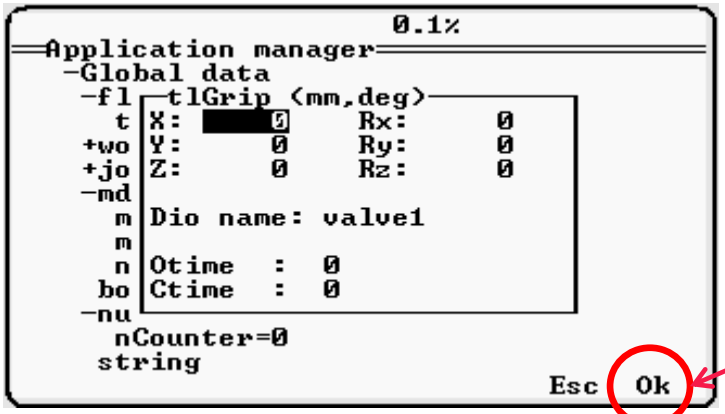
EDYCJA ZMIENNYCH



ENTER — Edycja zmiennej

ESC — Anuluje edycje

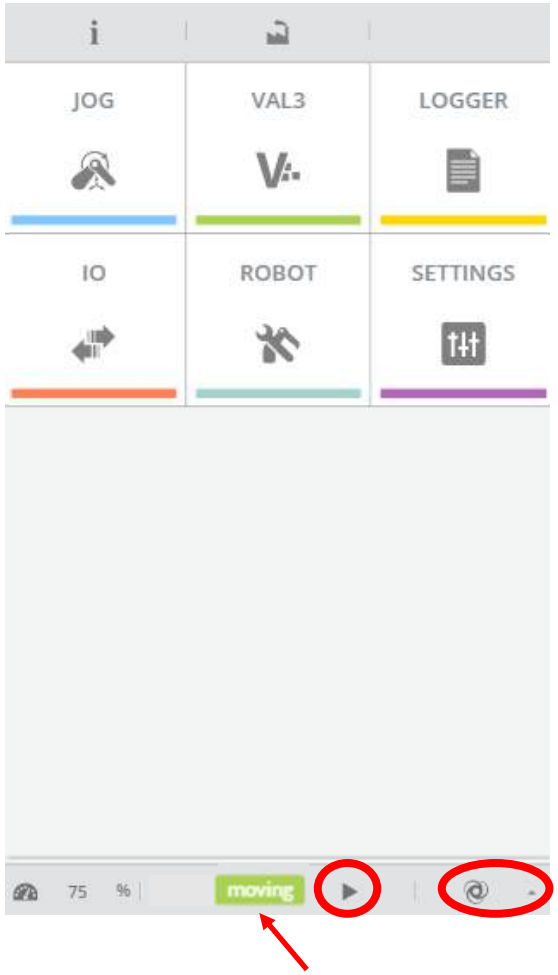
ENTER — Zatwierdza



W przypadku zmiennych złożonych zatwierdzenie następuje przez naciśnięcie przycisku OK)

ZEZWOLENIE NA RUCH TRYB AUTOMATYCZNY LOKALNY

1 Aplikacja jest uruchomiona, wybrano tryb automatyczny lokalny



2 Sprawdź stan układu bezpieczeństwa (E-Stop, zamknięte drzwi, stan czujników ,…)

3

Jak tylko zasilanie serwonapędów zostanie włączone,
Stan ramienia zmieni się na stalled
(Przycisk Move/hold zacznie mrugać)

Jeżeli ramię znajduje się w pozycji bazowej, naciśnięcie przycisku Move/hold uruchomi cykl. Stan ramienia zmieni się na: moving

Za każdym naciśnięciem przycisku Move/hold ruch ramienia zostaje wstrzymany. Status ramienia zmienia się na stalled Przycisk Move/hold zacznie mrugać

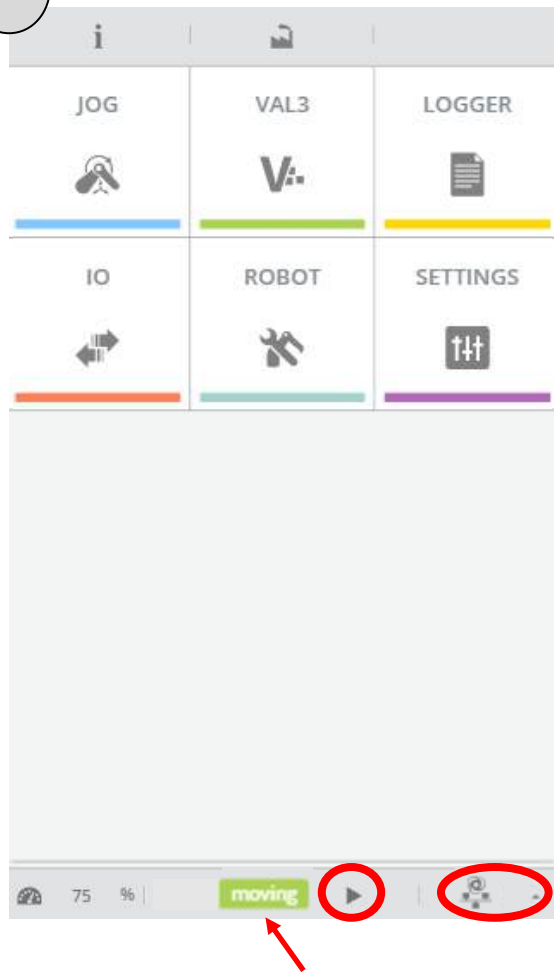
Każde wyłączenie zasilania ramienia zmienia jego status na connect ▶

Stan connect ▶ oznacza konieczność powrotu do punktu przerwania wykonywanej trajektorii

ZEZWOLENIE NA RUCH TRYB AUTOMATYCZNY ZDALNY

1

Aplikacja jest uruchomiona, wybrano tryb automatyczny zdalny



2

Sprawdź stan układu bezpieczeństwa (E-Stop, zamknięte drzwi, stan czujników, itp…)

3

Włączenie serwonapędów z teachpendanta jest niemożliwe. Zostaną one włączone przez zewnętrzne urządzenie:

- PLC lub inny sterownik.
- Teachpendant może zostać odłączony, a na jego miejsce wpięta zaślepka.

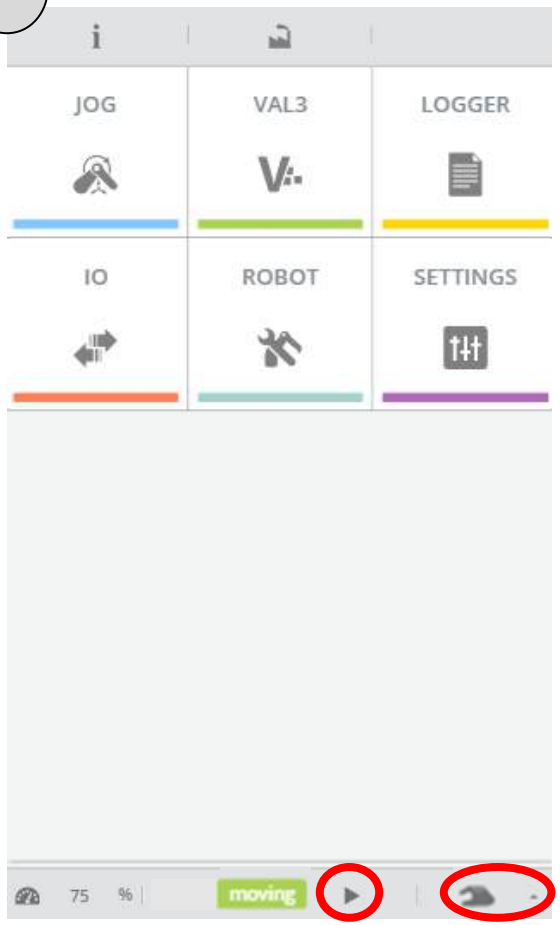
Jeżeli Teachpendant będzie odłączany, należy zapewnić inną możliwość zmiany prędkości robota i jego zatrzymania inną niż przycisk E-Stop

UWAGA: W zależności od programu, zaraz po włączeniu napędów, robot może wykonać ruch.

ZEZWOLENIE NA RUCH TRYB RĘCZNY

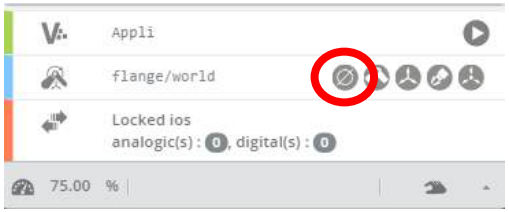
1

Aplikacja jest uruchomiona, wybrano tryb ręczny



2

Jeśli wybrany jest jeden tryb, przycisk przesuwania / wstrzymania nie miga. Aby dezaktywować tryb, wybierz «None»



Jak tylko zasilanie serwonapędów zostanie włączone, stan ramienia zmieni się na stalled
(Przycisk Move/hold zacznie mrugać)

Naciśnięcie przycisku Move/hold uruchomi cykl. Należy trzymać przycisk aby robot się poruszał. Stan ramienia zmieni się na moving

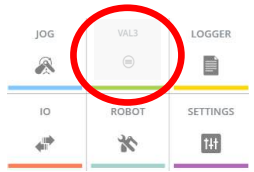
Każde puszczenie przycisku Move/hold zatrzymuje ruchy robota (stan ramienia zmienia się na stalled) – przycisk Move/hold zacznie mrugać

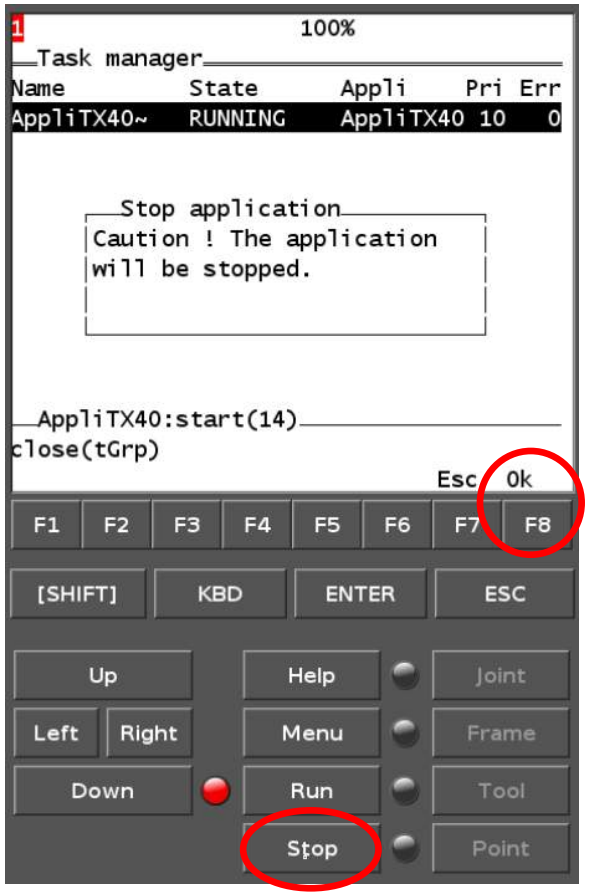
Stan connect ▶ oznacza konieczność powrotu do punktu przerwania wykonywanej trajektorii

ZATRZYMYWANIE APLIKACJI

1 Naciśnięcie przycisku dodatkowego menu

2 Naciśnięcie przycisku STOP





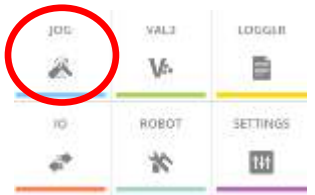
3 Potwierdzenie przyciskiem OK - F8

Uwaga: W programie STOP() można zaprogramować instrukcje wykonywane podczas zatrzymywania aplikacji (zerowanie, dodatkowe komunikaty itp.)

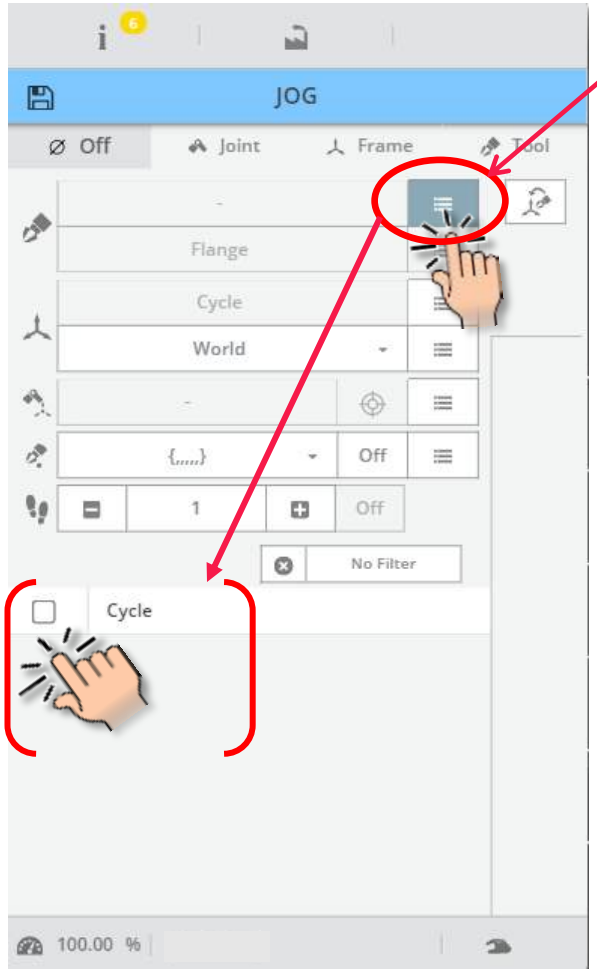
CZĘŚĆ 1 – OPERATOR CS9

NAUKA PUNKTÓW

DOSTĘP DO LISTY PUNKTÓW W MENU JOG (1/2)

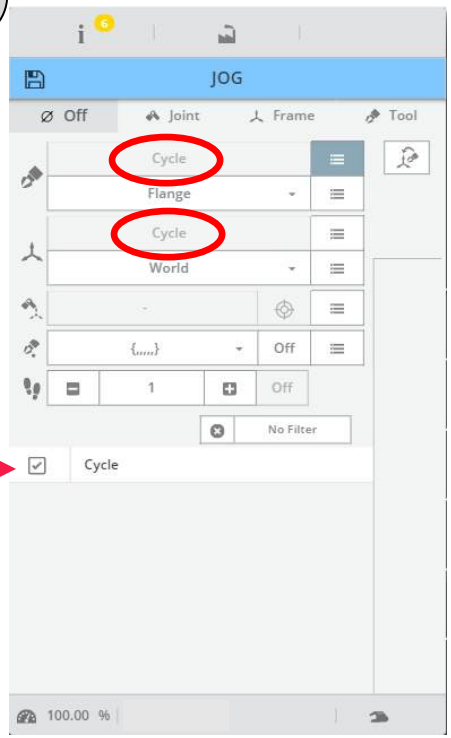


1 Wybór aplikacji za pomocą APPLICATION MANAGER (slajd 36)



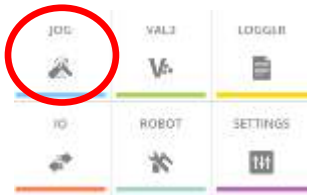
2 Wybór aplikacji w menu JOG osobno dla narzędzia i układu współrzędnych

3

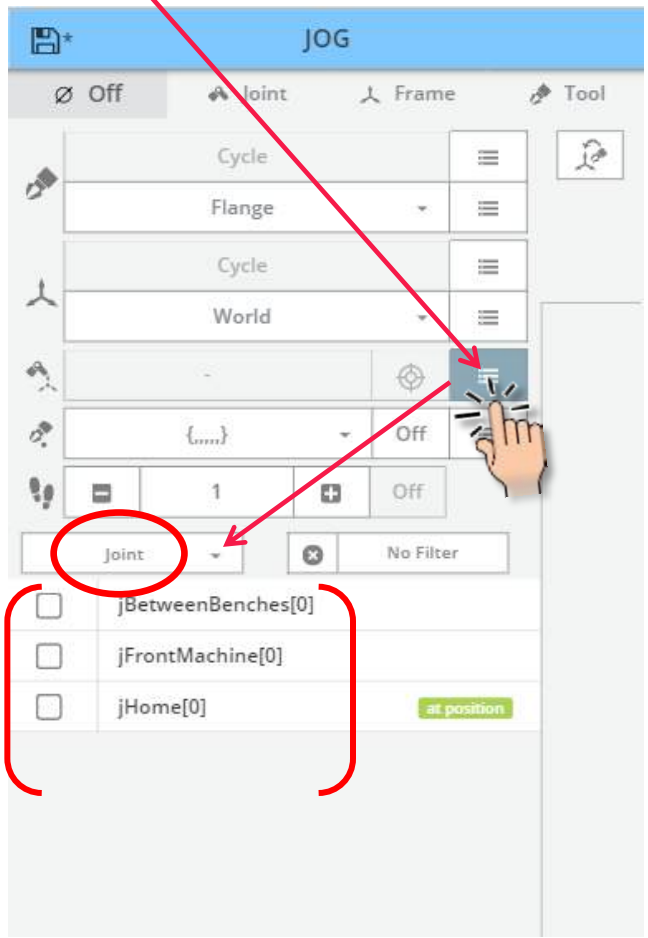


Nazwa aplikacji pojawia się w menu TOOL i FRAME

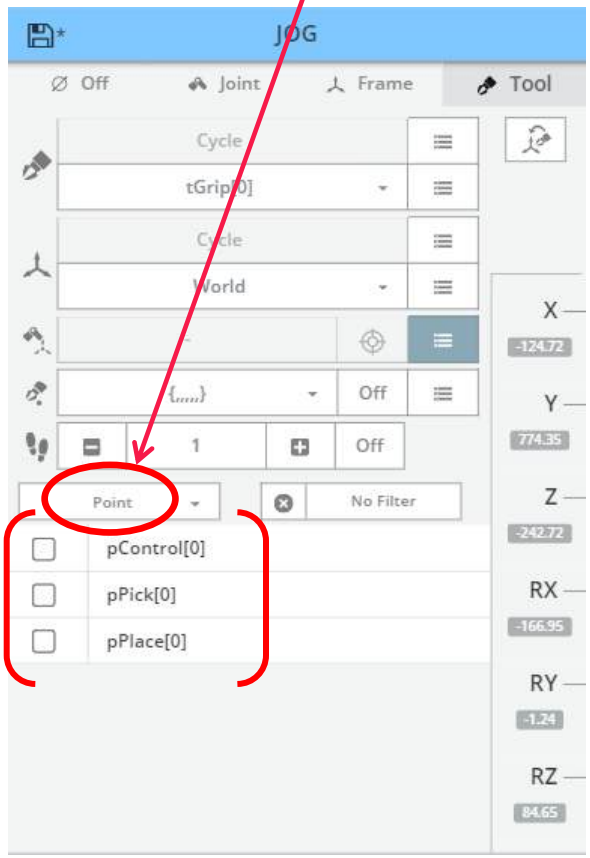
DOSTĘP DO LISTY PUNKTÓW W MENU JOG (2/2)



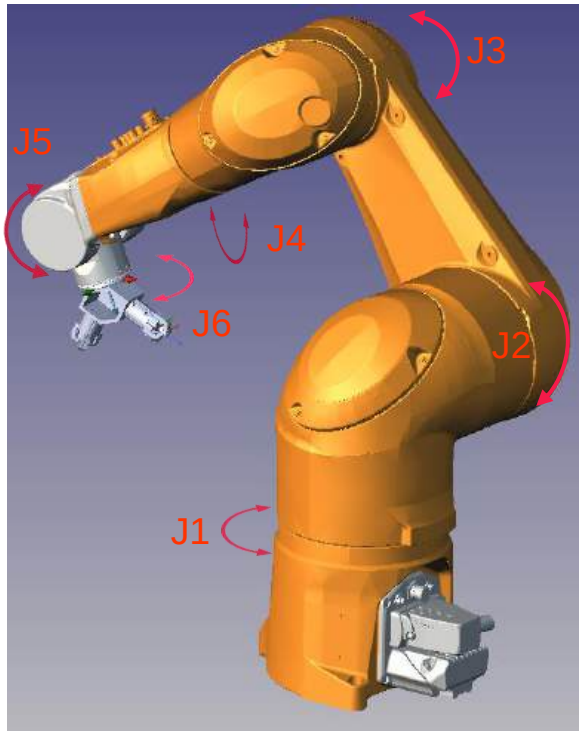
4 Naciśnięcie menu wyboru punktów



2 rodzaje punktów (**JOINT** oraz **POINT**)



PUNKT TYPU: JOINT



Przykłady nazw: jDepart, jHome, jApproach

- Współrzędne punktu przechowują położenie kątowe wszystkich osi robota: J1,….J6
- Staubli zaleca aby nazwy tego rodzaju punktów zaczynać od małej litery j
- Punkt wymusza konfigurację ramienia:

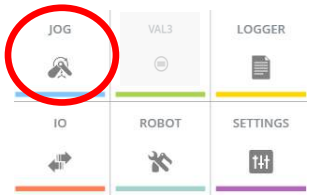
Przykład:

- Ramię po prawej stronie przewodów
- Łokieć powyżej osi 2
- Nadgarstek ustawiony tak że przyłącza pneumatyczne są na górze

Wykorzystane w przypadku:

- Unikanie kolizji z oprzyrządowaniem
- Unikanie punktów osobliwych

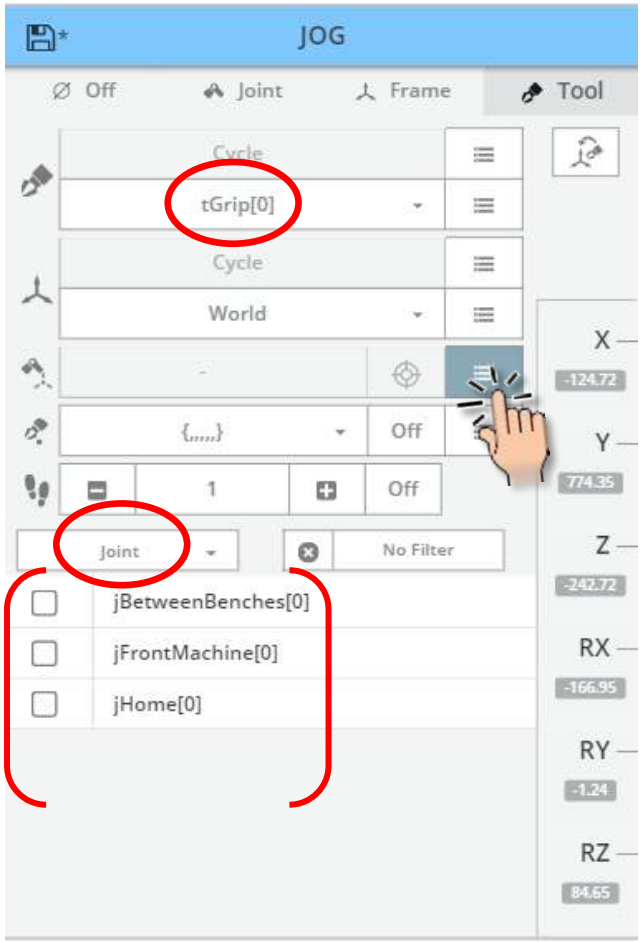
NAUKA PUNKTU JOINT (1/2)

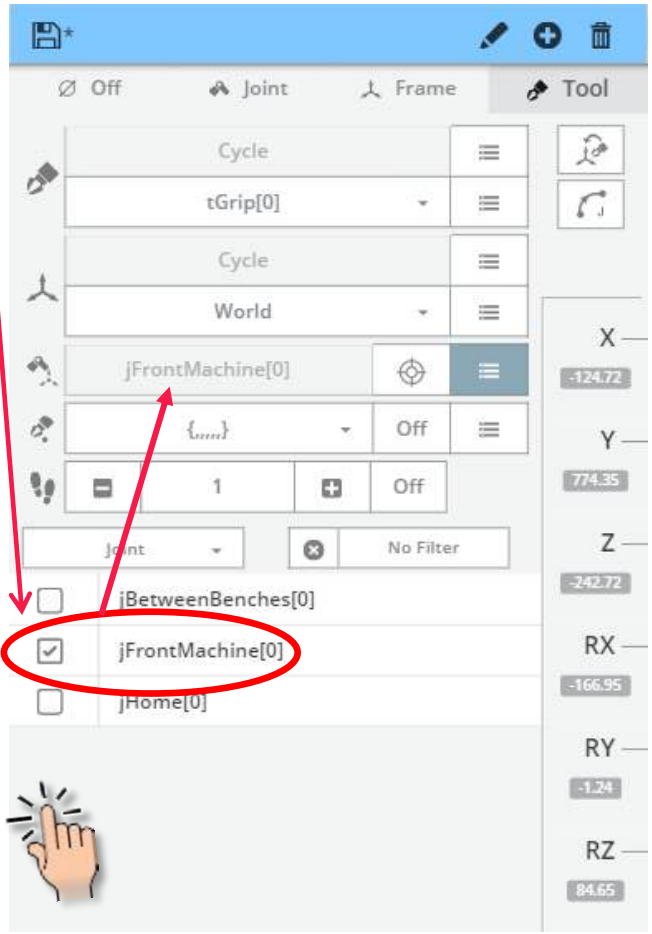


1 Ustaw ramie w pożądanej pozycji, wykorzystując ręczny trym ruchu i menu JOG

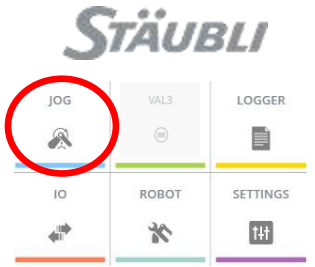
2 Wybierz odpowiednie narzędzie

3 Wybierz przeuczany punkt

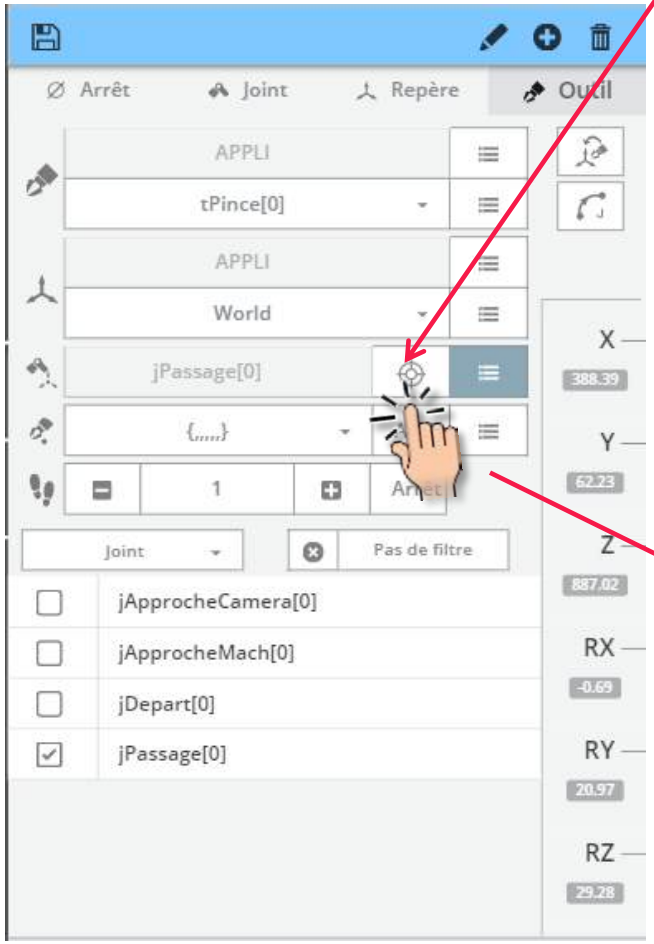


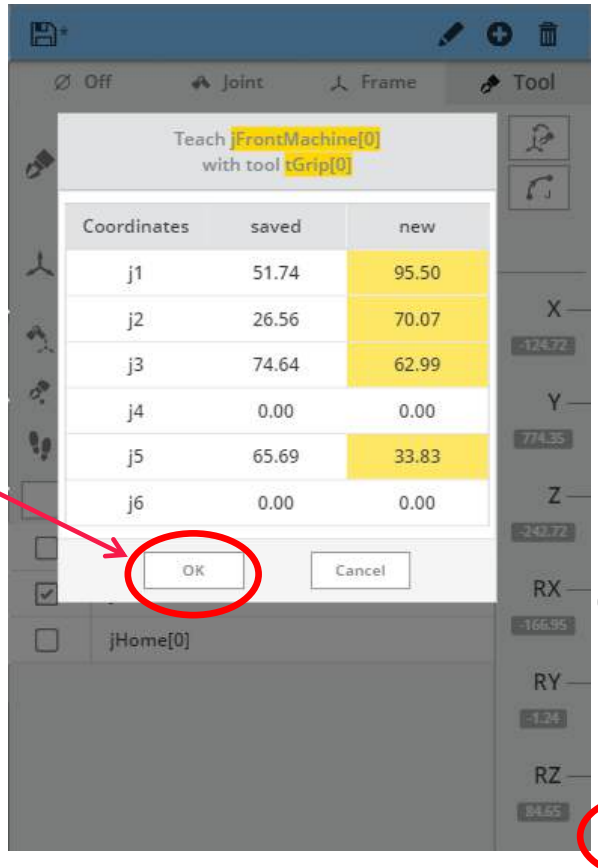


NAUKA PUNKTU JOINT (2/2)



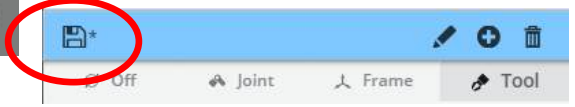
4 Naciśnij przycisk nauki punkt i potwierdź przyciskiem OK



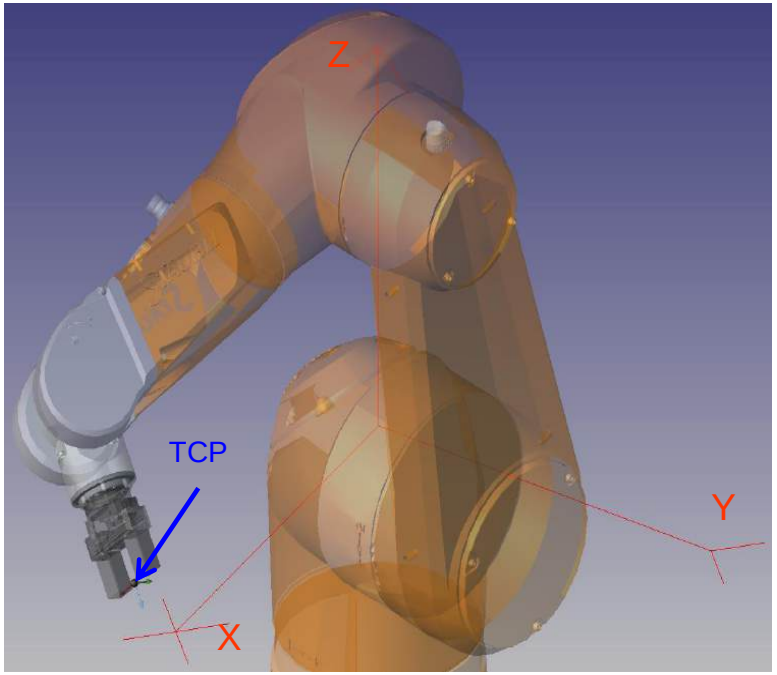


Obok nazwy punktu pojawi się informacja o tym że robot osiągnął żądaną pozycje

5 **Nie zapomnij zapisać zmian**

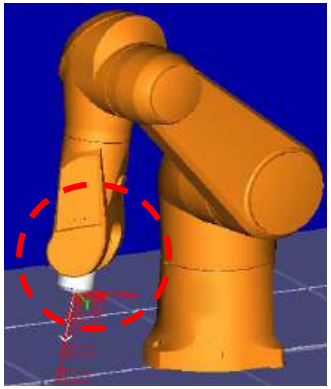


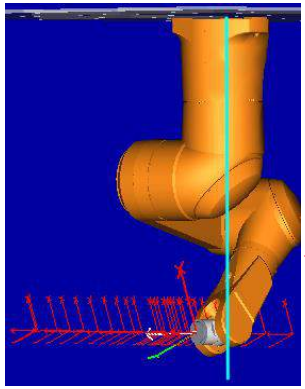
PUNKT TYPU: POINT



Przykłady nazw : pPick, pPlace, pTraj[ ]

- Współrzędne punktu przechowują pozycje X,Y,Z w mm oraz nachylanie RX,RY,RZ w stopniach punktu TCP w wybranym układzie współrzędnych
- Staubli zaleca oby nazwy tego rodzaju punktów zaczynać od małej litery p
- Konfiguracja ramienia w danym punkcie może się zmieniać



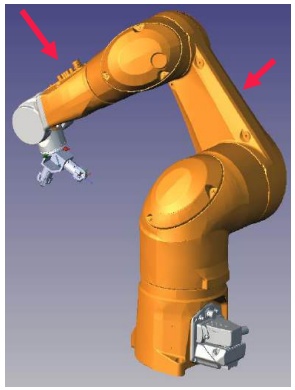


Używane do:

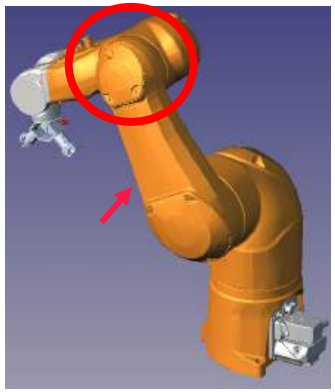
- Zapisania pozycji w układach współrzędnych (paletyzacja, skomplikowane trajektorie)
- Uwaga na możliwość wystąpienia punktów osobliwych (pozycja osi 5 w pobliżu 0 stopni, nadgarstek w pobliżu osi 1)

DO 8 RÓŻNYCH KONFIGURACJI DLA JEDNEGO PUNKTU POINT

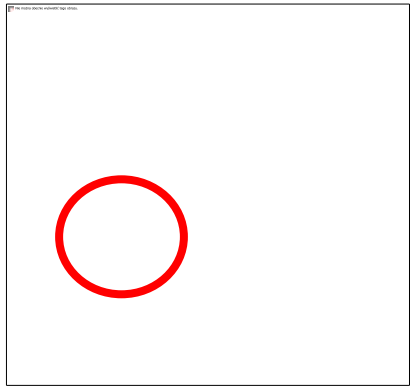
2 konfiguracje ramienia.



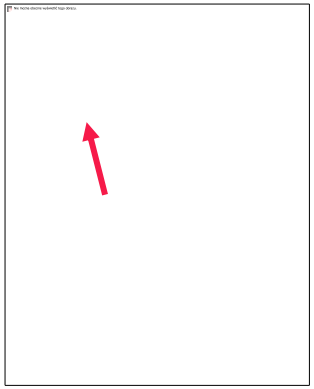
2 konfiguracje łokcia



2 konfiguracje nadgarstka



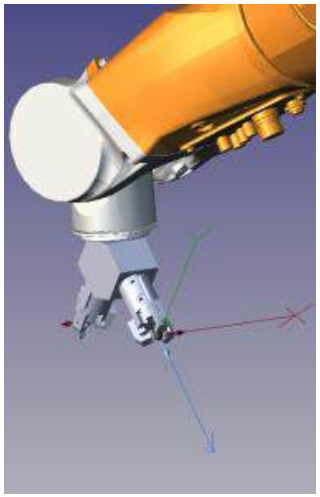
2 x 2 x 2 kombinacji



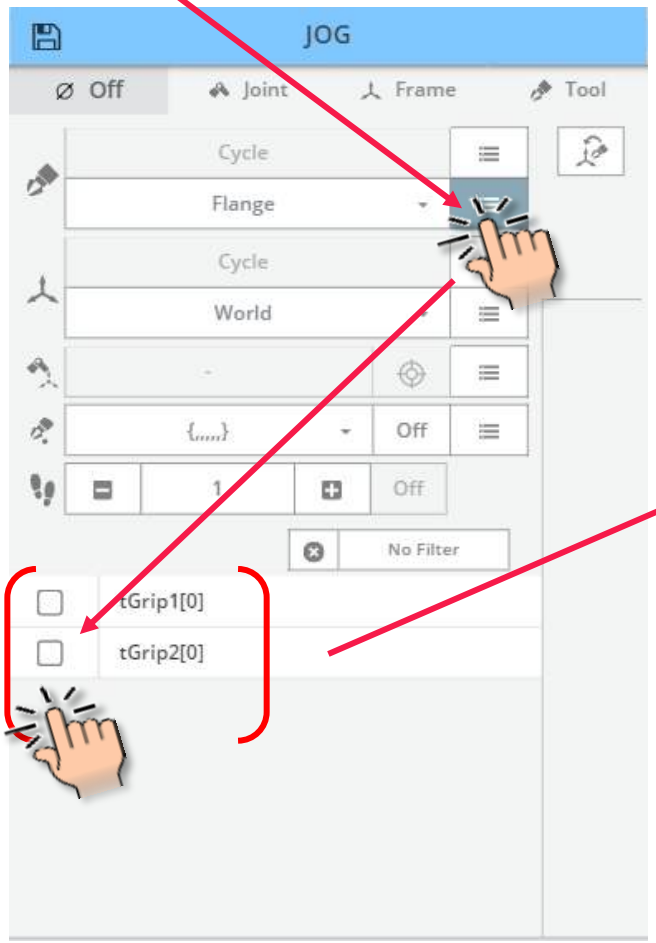
Wymuszamy konfigurację ramienia w punkcie JOINT

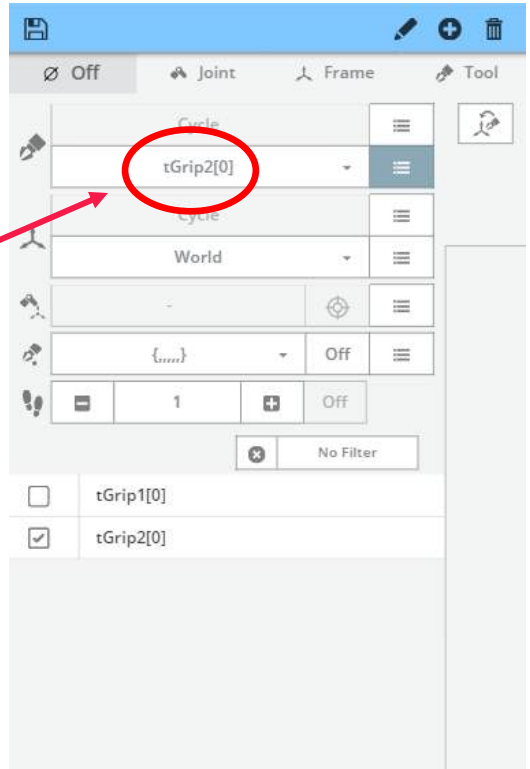
- Przejazdy (niedokładne) po punktach typu: JOINT (szybka zmiana pozycji bez zatrzymywania)
- Dokładne dojazdy do punktów (pobieranie, odkładanie,) typu: POINT
- Zaprogramuj co najmniej 1 punkt JOINT w całej trajektorii
- Jeżeli zachodzi konieczność zmiany konfiguracji ramienia, można to zrealizować poprzez przejazd przez punkty Joint.

WYBÓR NARZĘDZIA



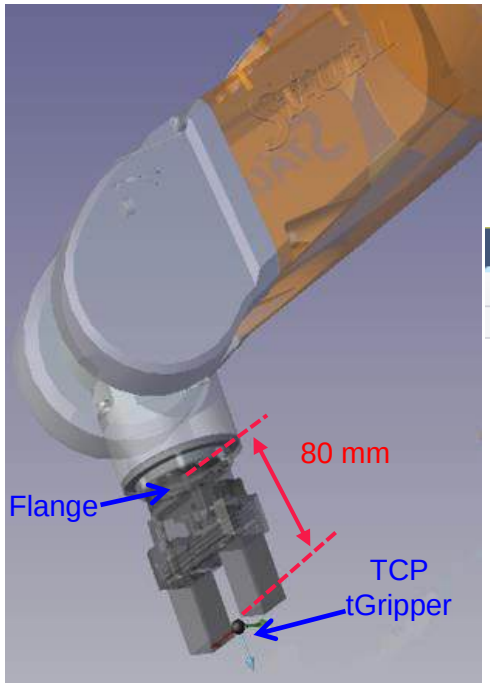
1





Należy dokonać wyboru odpowiedniego narzędzia przed dojazdem oraz nauką punktów kartezjańskich!

ZALEŻNOŚĆ POMIĘDZY PUNKTAMI KARTEZJAŃSKIMI I NARZĘDZIAMI



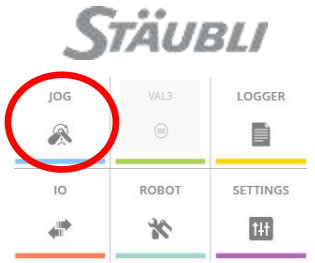
3D View | Cycle-tGripper Data*

Index	X	Y	Z	Rx	Ry	Rz	Gripper	OTime	CTime
0	0	0	80	0	0	0	DsiIO\Qvalve1-1	0	0

Zdefiniowanie geometrii narzędzia pozwala na:

- Dojazd do tych samych punktów z różnymi narzędziami
- Kontrolę prędkości końcówek roboczych narzędzi
- Zmianę geometrii narzędzia bez konieczności przeuczania punktów
- Łatwe nauczanie punktów przez ustalenie pozycji i orientacji wokół punktu TCP (Tool Center Point)

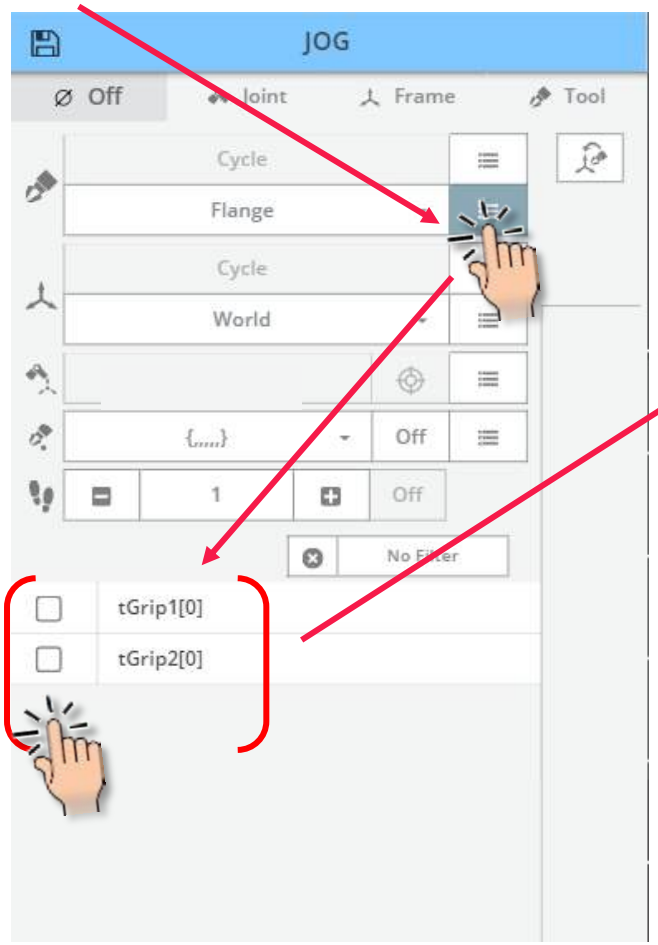
NAUKA PUNKTU POINT (1/2)

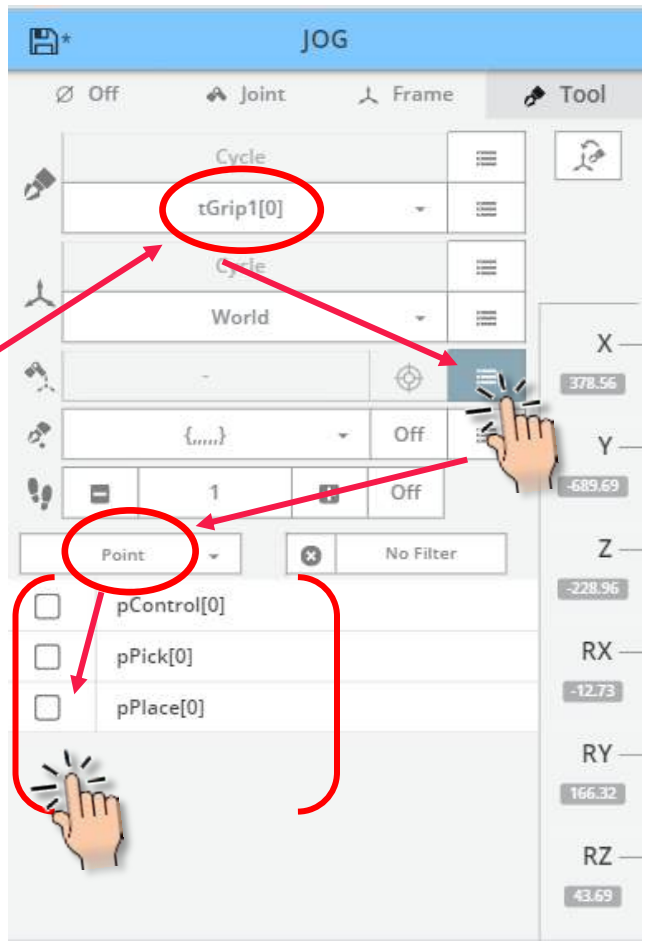


1 Ustaw ramie w pożądanej pozycji, wykorzystując ręczny trym ruchu i menu JOG

2 Wybierz odpowiednie narzędzie

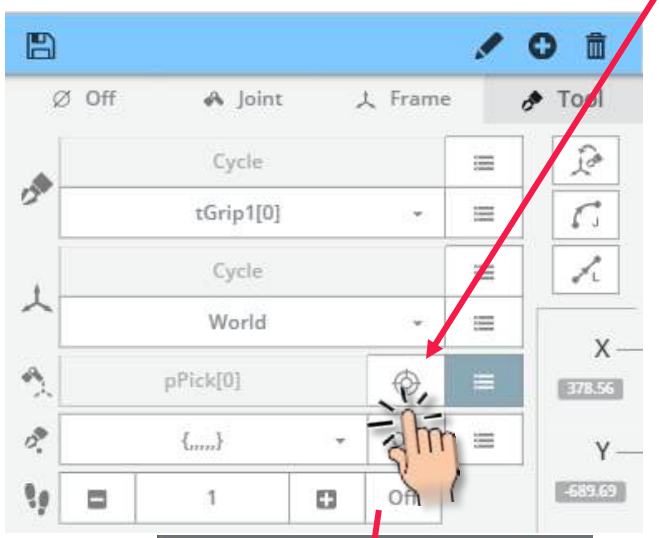
3 Wybierz przeuczany punkt

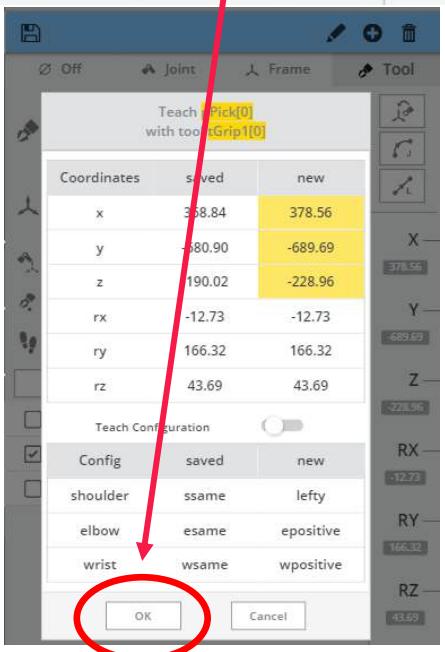




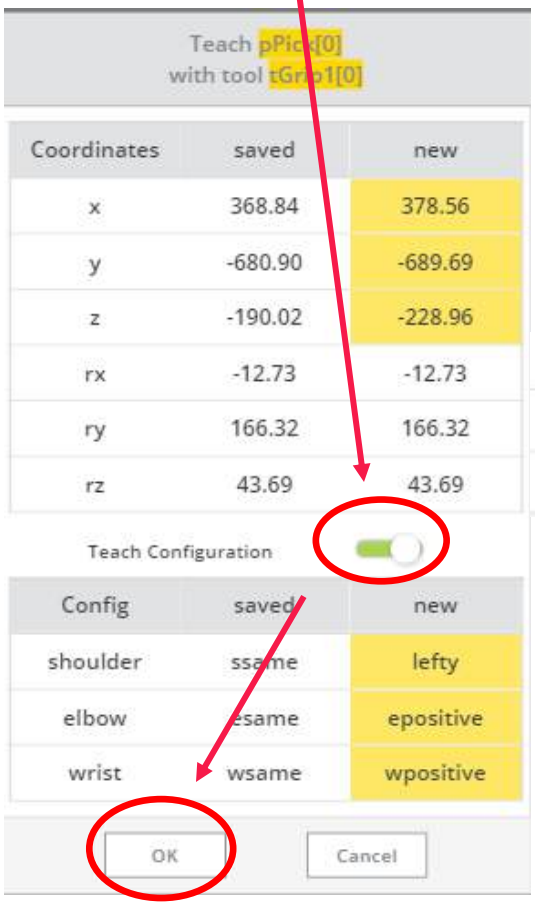
NAUKA PUNKTU POINT (2/2)

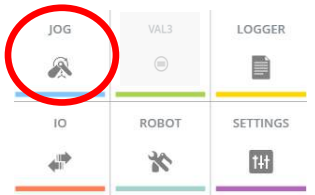
4 Naciśnij przycisk nauki punkt i potwierdź przyciskiem OK





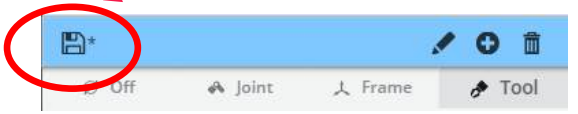
Nauka konfiguracji ramienia, wymusza jego pozycję



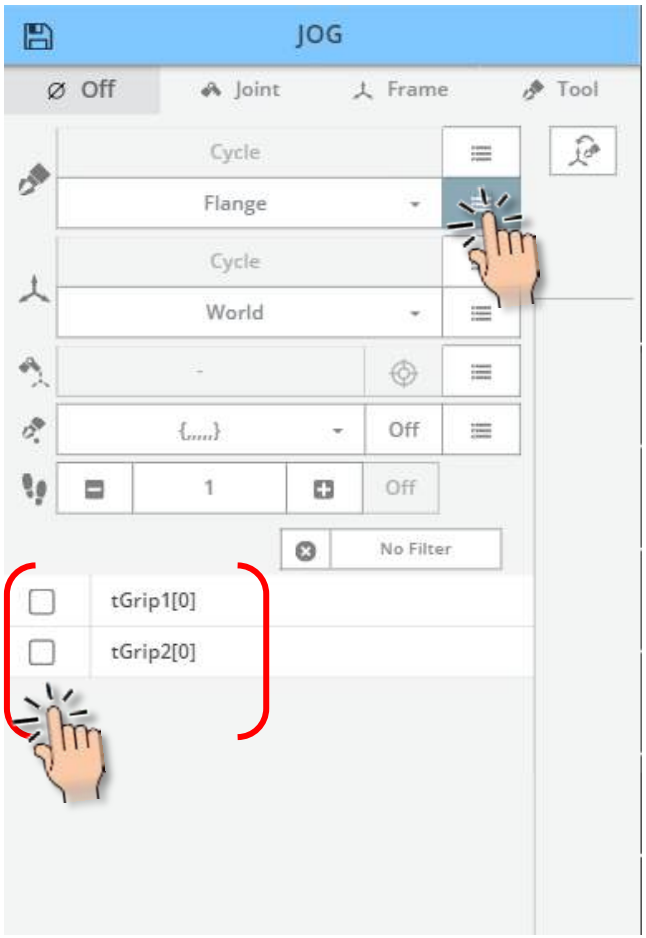


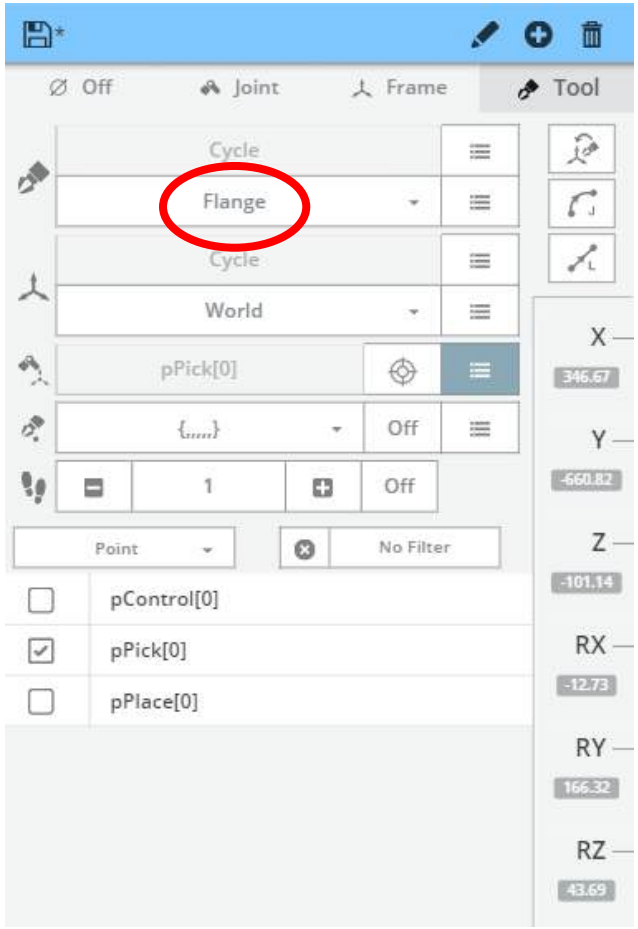
Obok nazwy punktu pojawi się informacja o tym że robot osiągnął żądaną pozycje

5 **Nie zapomnij zapisać zmian**



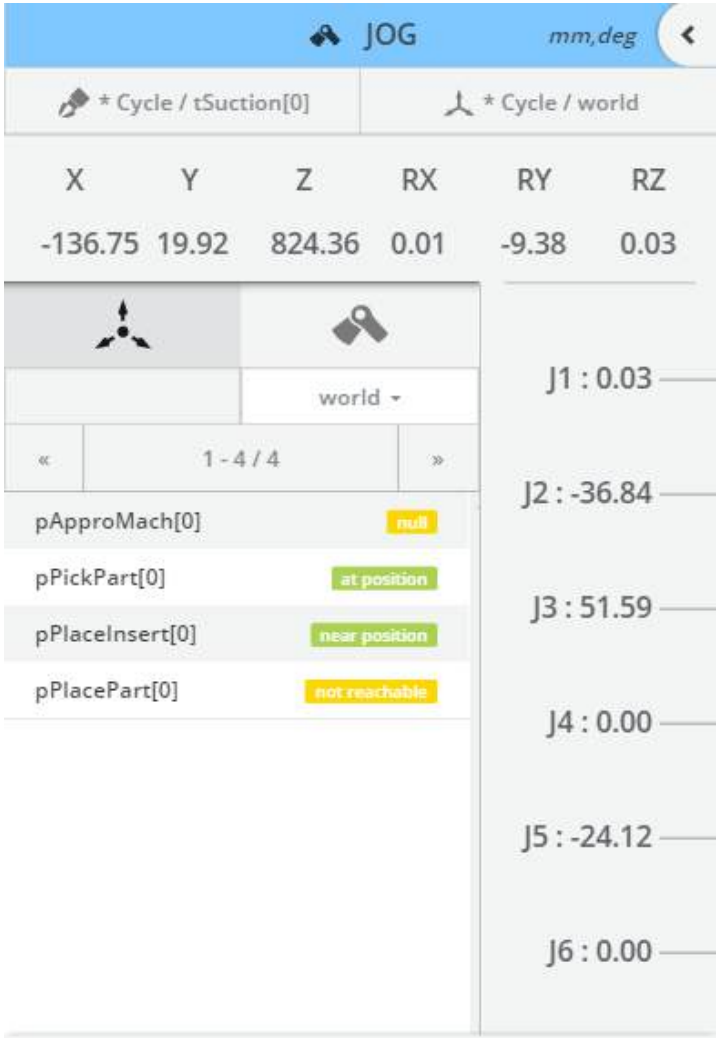
MOŻLIWA POMYŁKA





Podczas nauki punktów wybierz odpowiednie narzędzie – nie « flange » środek flanszy osi 6

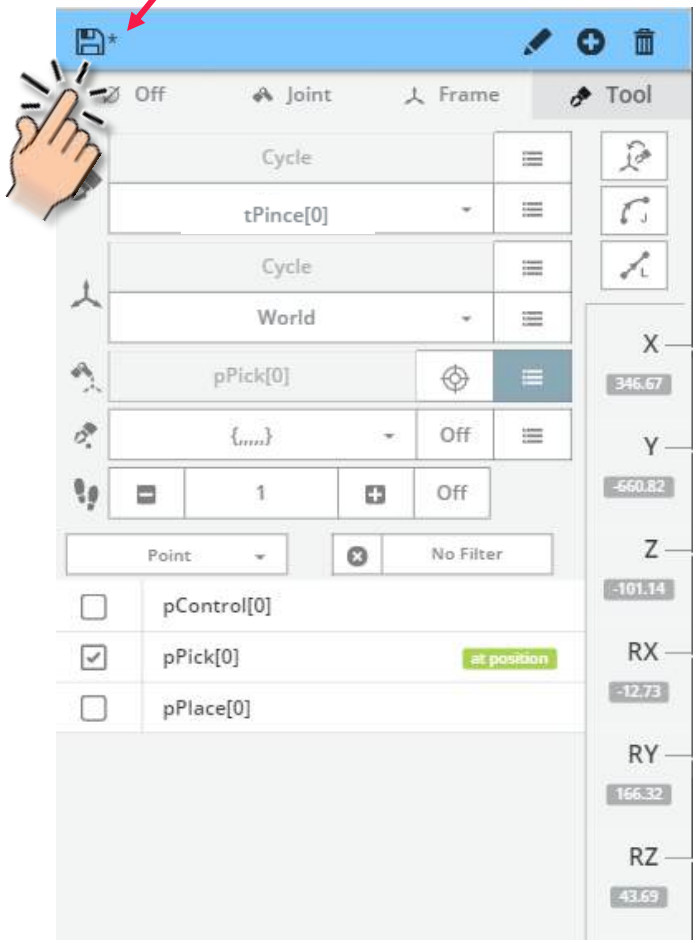
STATUS PUNKTU



- null Punkt nie nauczony, wszystkie współrzędne = 0
- Robot na pozycji z wybranym narzędziem (około 0.01 mm)
- Robot w pobliżu punktu (około 1 mm)
- Pozycja nie osiągalna z wybranym narzędziem

ZAPISYWANIE

Jeżeli nie zapisano aplikacji lub punktów, obok ikony zapisu pojawia się gwiazdka



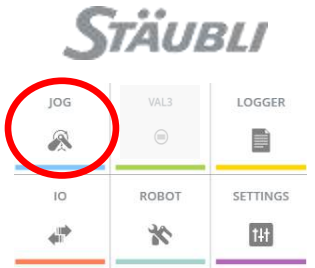
1 Aby zapisać zmiany, naciśnij ikonę dyskietki

CZĘŚĆ 1 – OPERATOR CS9

POINT MODE – DOJAZD DO PUNKTÓW

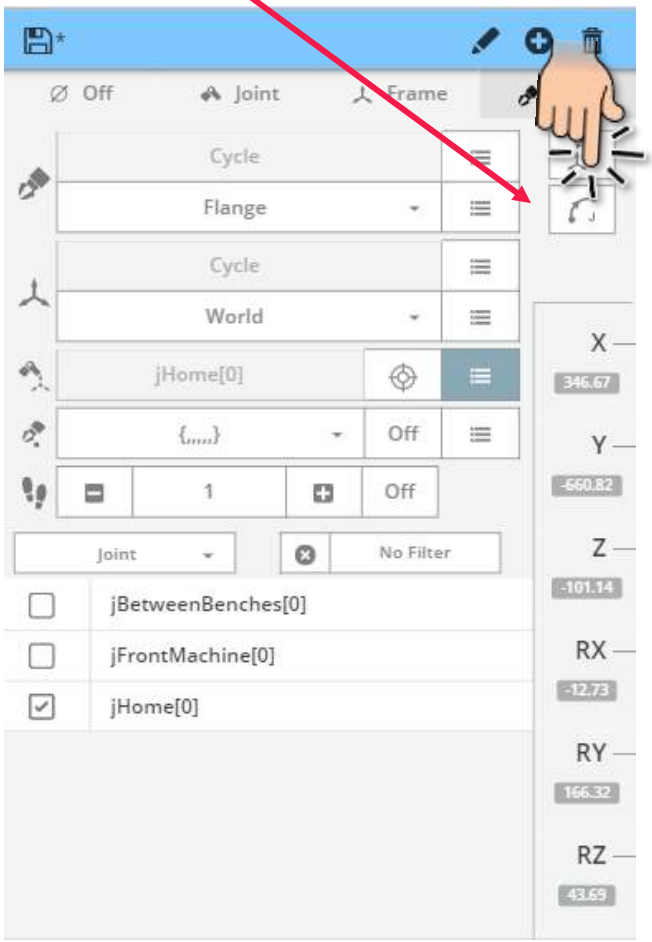
TRYB RUCHU JOINT

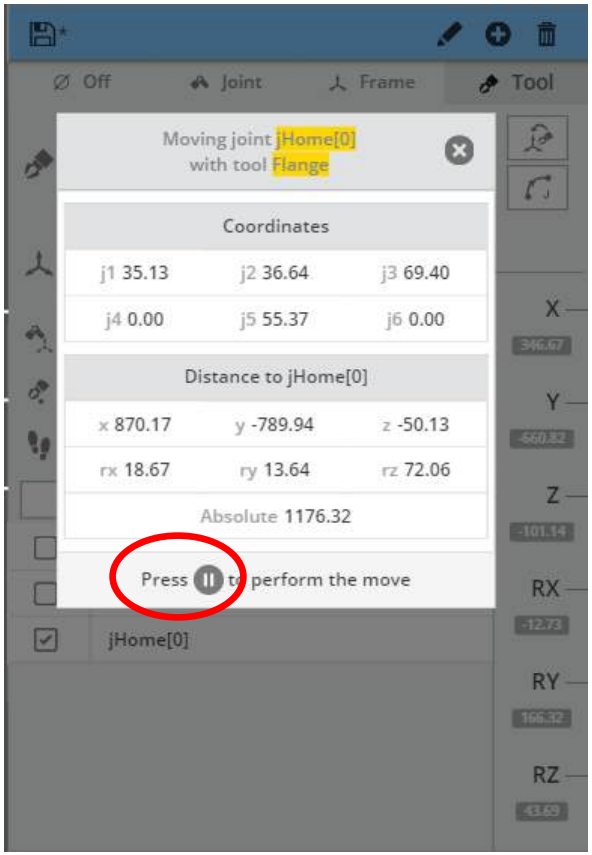
Punkty JOINT



1 Naciśnięcie przycisku  pozwala na dojazd w trybie JOINT (po krzywej) do danego punktu

2 Naciśnij i przytrzymaj przycisk Move/Hold, aby dojechać do punktu



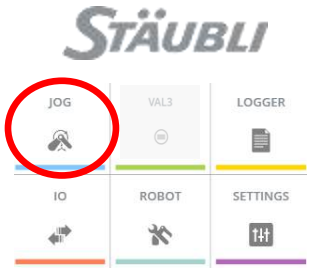


Po dojechaniu do punktu, obok jego nazwy pojawi się informacja at position

UWAGA NA KOLIZJE
ramię wykona ruch po najkrótszej drodze do punktu

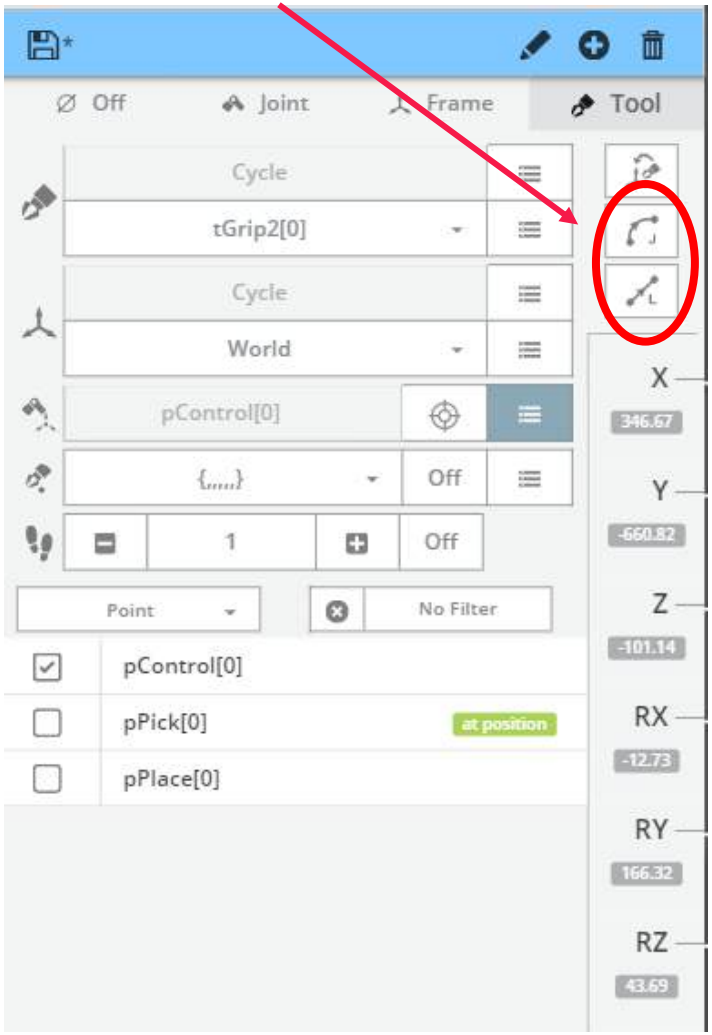
TRYB RUCHU POINT

Punkty POINT



1

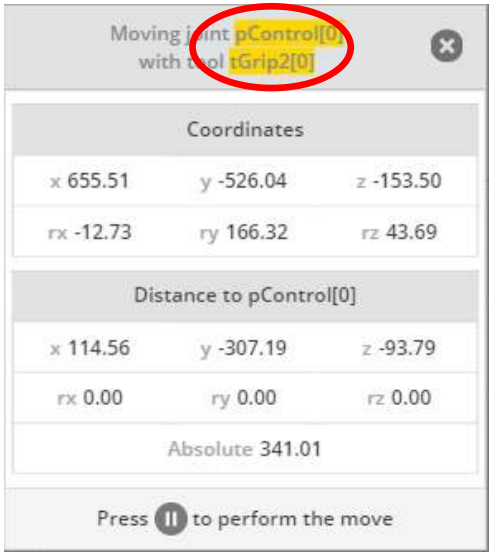
Przyciski  i  pozwalają na dojazd w trybie JOINT (po krzywej) lub LINE (po prostej) do wybranych punktów



2

Naciśnij i przytrzymaj przycisk Move/Hold, aby dojechać do punktu

Wybierz odpowiednie narzędzie



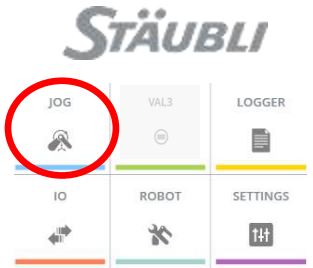
Po dojechaniu do punktu, obok jego nazwy pojawi się informacja at position

UWAGA NA KOLIZJE ramię wykona ruch po najkrótszej drodze do punktu

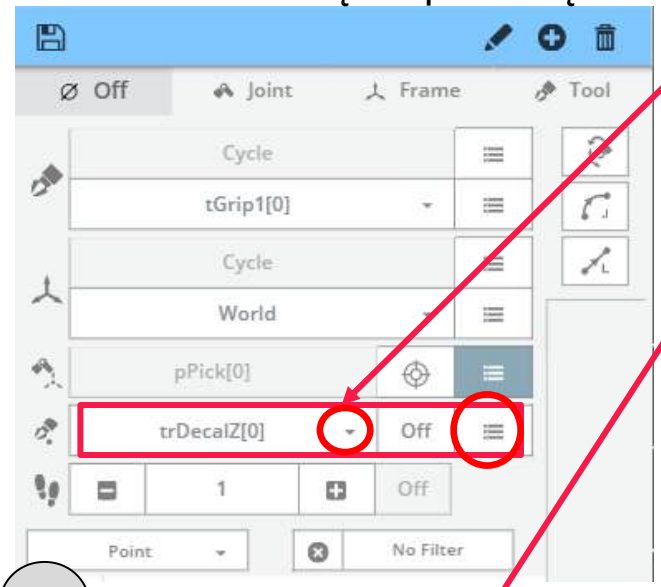
TRYB RUCHU POINT – DOJAZD NAD PUNKTY

POINT (1/2)

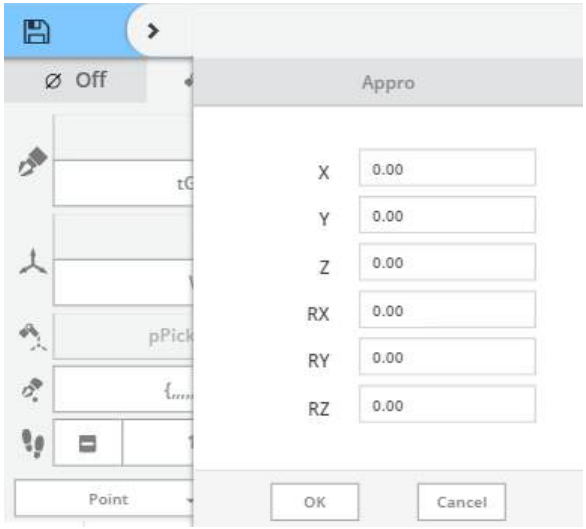
Tylko dla punktów POINT



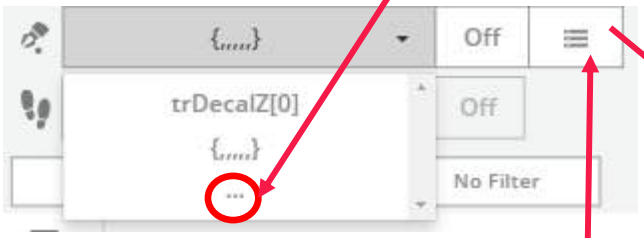
1 Aby dojechać nad punkt **POINT**:
Rozwiń listę za pomocą strzałki znajdującej się obok ikony



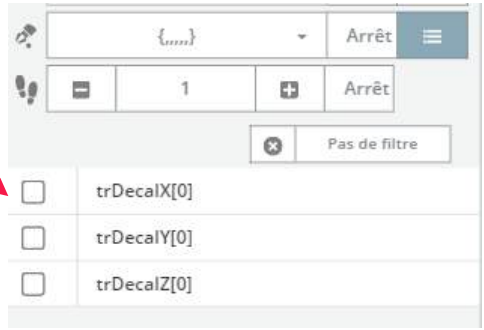
Wybranie opcji „…” pozwala na wpisanie wartości odległości dojazdu do punktu



2 Lista wyświetla ostatnio używane wartości



Naciśnięcie przycisku  wyświetla pozostałe wartości

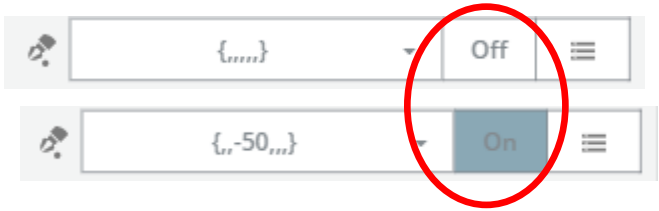


TRYB RUCHU POINT – DOJAZD NAD PUNKTY POINT (2/2)

Tylko dla punktów POINT

3

Aktywacja/dezaktywacja dojazdu nad punkty POINT odbywa się przez naciśnięcie przycisku

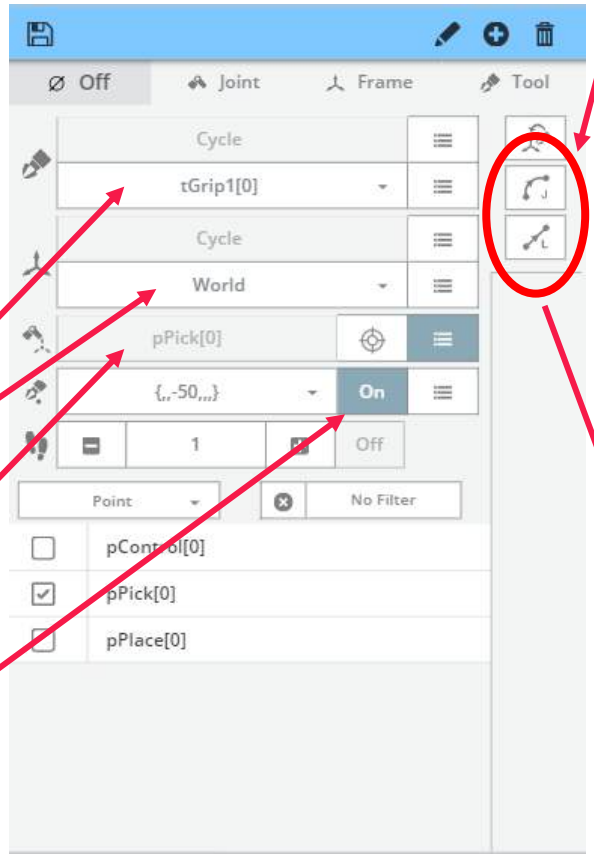


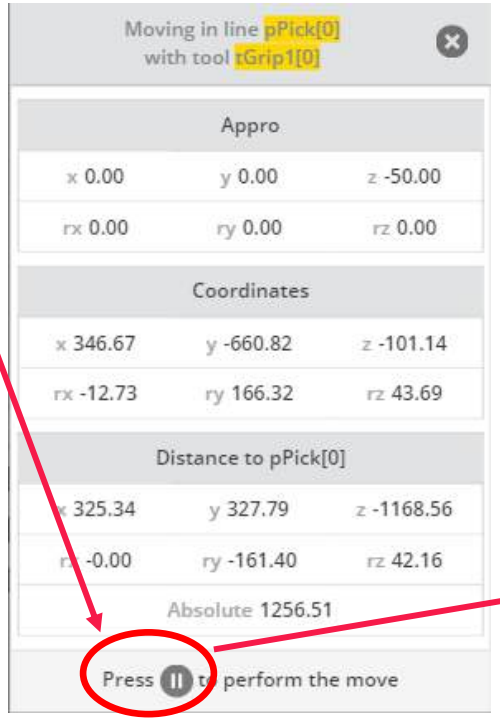
4

Wybranie rodzaju ruchu (po krzywej lub prostej) nad punkt POINT

Na jednej stronie menu, wyświetlono wszystkie parametry ruchu ręcznego:

- Aktualne narzędzie
- Aktualny układ współrzędnych
- Wybrany punkt
- Aktywacje dojazdu nad punkt





CZĘŚĆ 1 – OPERATOR CS9

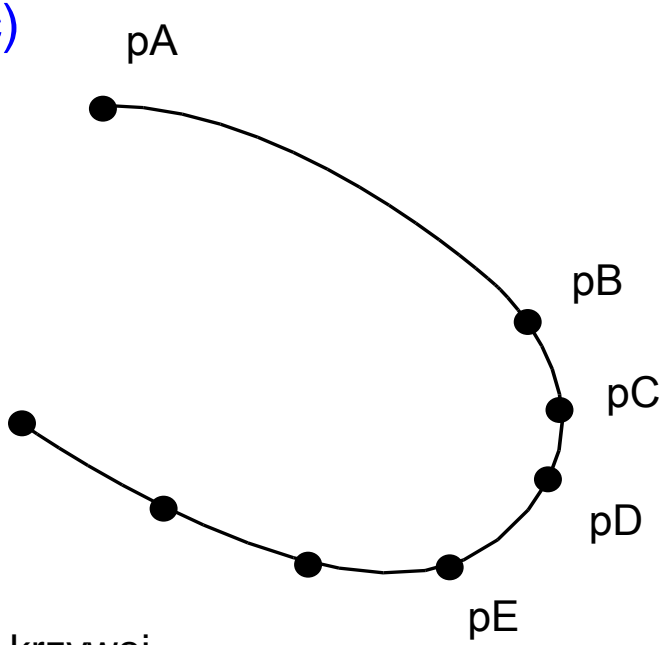
INSTRUKCJE RUCHU

RUCH PO LINII KRZYWEJ: MOVEJ

Movej(point,tool,mdesc)

lub

Movej(joint,tool,mdesc)



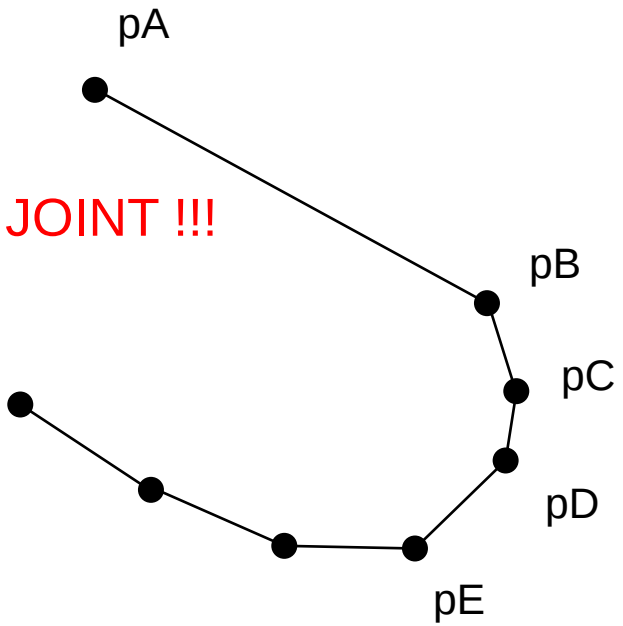
Movej(pA,tGrip,mFast)
Movej(pB,tGrip,mFast)
Movej(pC,tGrip,mFast)
Movej(pD,tGrip,mFast)
Movej(pE,tGrip,mFast)
....

- Interpolacja Joint: ruch po krzywej
- Prędkość i przyspieszenia zapisane w zmiennych motion descriptor (Mdesc)
- Brak punktów osobliwych
- Tryb wykorzystywany gdy nie występują: przeszkody, pobieranie, odkładanie,...

RUCH PO LINII PROSTEJ: MOVEL

Movel(point,tool,mdesc)

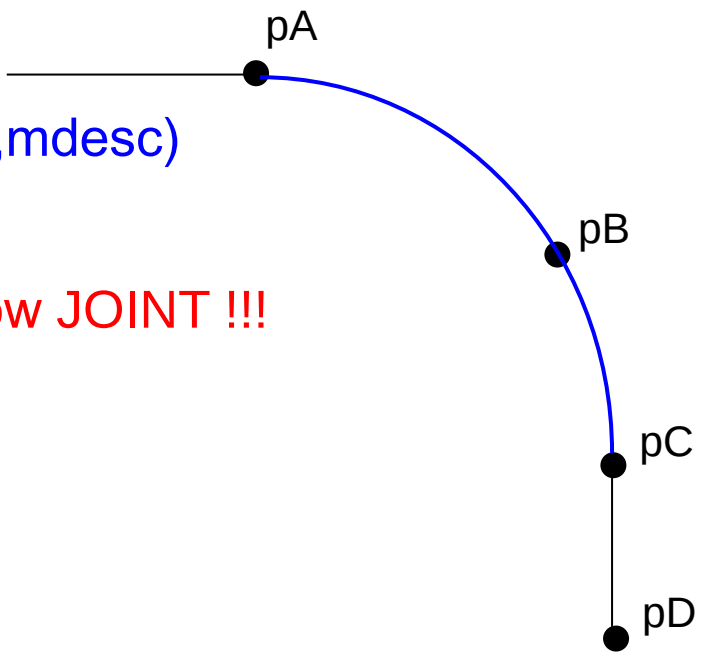
Niemożliwe dla punktów JOINT !!!



Movel(pA,tGrip,mFast)
Movel(pB,tGrip,mFast)
Movel(pC,tGrip,mFast)
Movel(pD,tGrip,mFast)
Movel(pE,tGrip,mFast)
....

- Interpolacja Kartezjańska: ruch po linii prostej
- Prędkość i przyspieszenia zapisane w zmiennych motion descriptor (Mdesc)
- Występują punkty osobliwe => mniejsza prędkość ruchu
- Tryb wykorzystywany gdy występują: przeszkody, pobieranie, odkładanie,...

RUCH PO ŁUKU: MOVEC



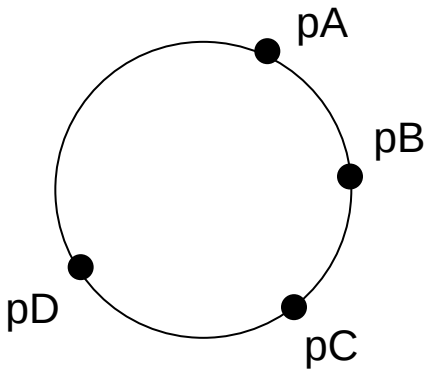
Movec(point,point,tool,mdesc)

Niemożliwe dla punktów JOINT !!!

Movel(pA,tGrip,mFast)

Movec(pB,pC,tGrip,mFast)

Movel(pD,tGrip,mFast)

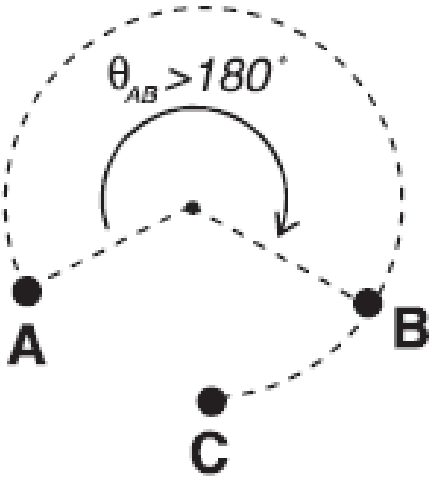


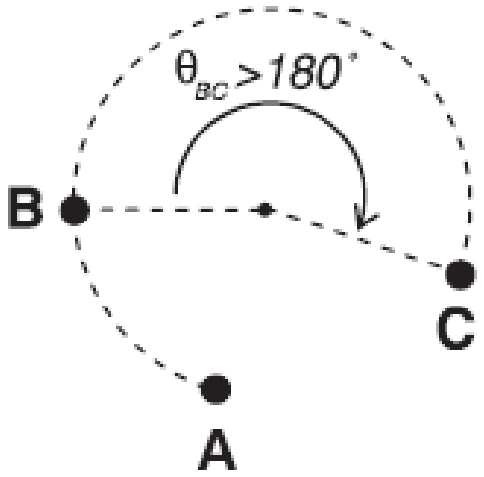
- Interpolacja kołowa: ruch po łuku
- Okrąg można zaprogramować za pomocą 4 punktów:
  - Movec(pB,pC,tGrip,mFast)
  - Movec(pD,pA,tGrip,mFast)

OGRANICZENIA: MOVEC

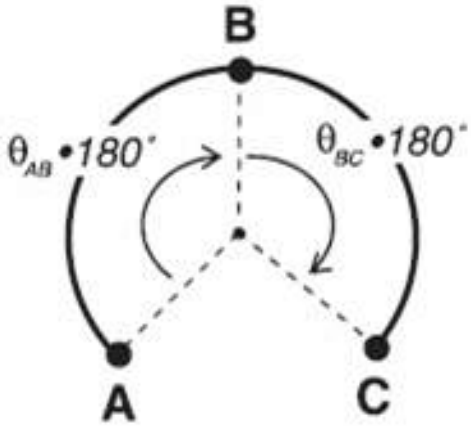
Niedozwolone ruchy

(brak rozwiązania)





Punkt pośredni musi tak umieszczony aby kąt pomiędzy nim z punktami początkowym i końcowym nie był większy niż 180 °



APLIKACJA (PROGRAM) A RUCH RAMIENIA

Wykonywanie programu

Ruchy robota

movej(pA, ,)
movel(pB , ,)
movej(pC , ,)
movel(pD , ,)

movej, movel, movej,...

MOVEMENT STACK

pD
pC
pB
pA

Przycisk Move/Hold

!!! NIEZALEŻNOŚĆ WYKONYWANIA PROGRAMU I RUCHU ROBOTA!!!

W niektórych przypadkach, aplikacja (program) musi zostać zatrzymany, przeładowany i uruchomiony ponownie, aby zmiany w punktach zostały uwzględnione.

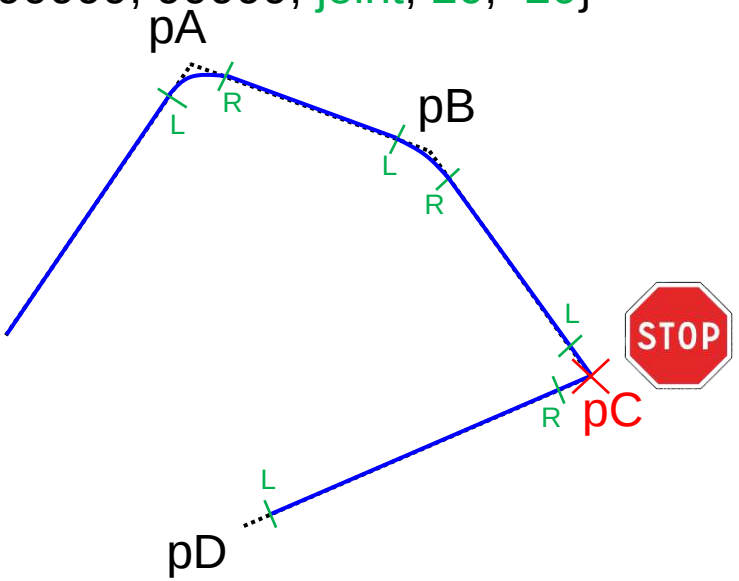
SYNCHRONIZACJA RUCHU I APLIKACJI (PROGRAMU)

- Instrukcja **waitEndMove()** pozwala na wstrzymanie wykonywania programu do momentu osiągnięcia fizycznie danego punktu przez ramię robota
- Po dojechaniu do punktu robot się zatrzyma – nastąpi wyłączenie wygładzania trajektorii (blending)

ACCEL VEL DECEL TVEL RVEL BLEND LEAVE REACH

- Przykład: mRapide = {100, 100, 100, 99999, 99999, joint, 20, 20}

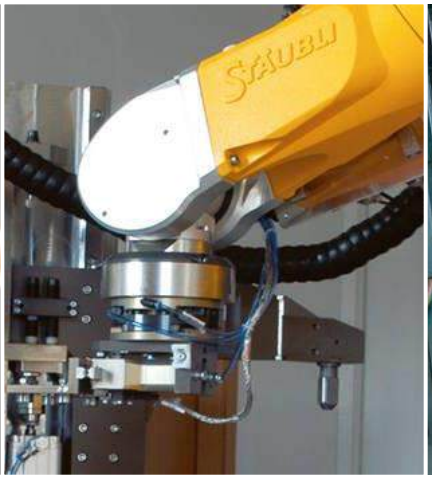
```
movel(pA,tGripper,mFast)
movel(pB,tGripper,mFast)
movel(pC,tGripper,mFast)
waitEndMove()
movel(pD,tGripper,mFast)
```



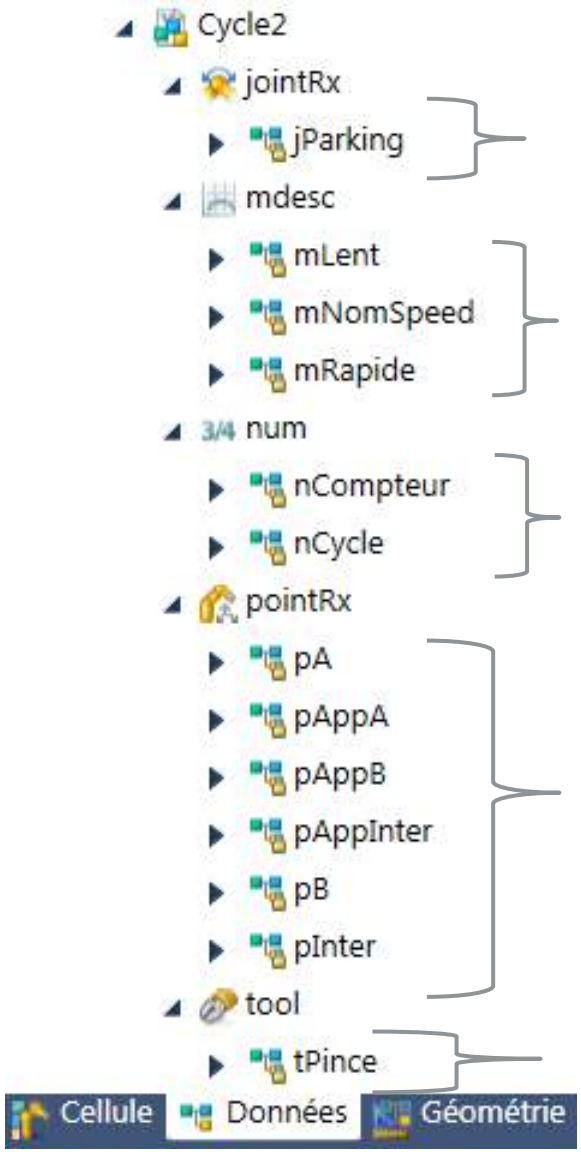
- waitEndMove() wykorzystujemy tylko gdy robot musi się zatrzymać w punkcie

- CZĘŚĆ 1 – OPERATOR CS9

GLOBAL DATA


GLOBAL DATA



Pozycje kątowe, punkty typu **jointRX**

- Pozycja względem zera mechanicznego (0, 0, 0, 0, 0, 0)

Parametry prędkości, **mdesc**

- Prędkość, przyspieszenie, hamowanie, blending (on/off), prędkość liniowa, prędkość kątowa

Zmienne numeryczne, **num**

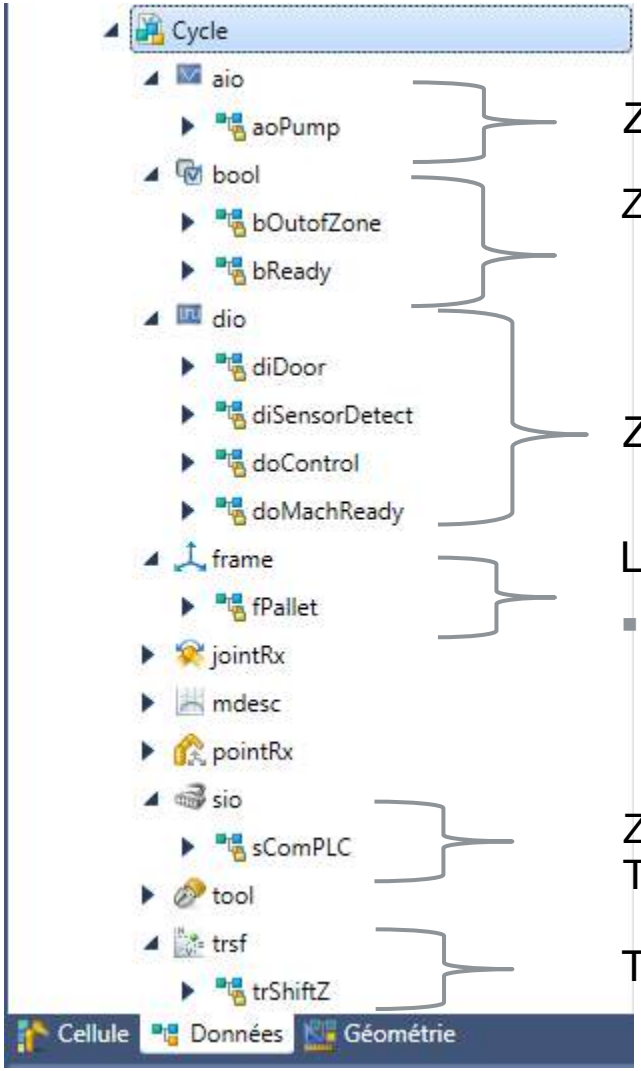
- nIndex=20
- Operacje na zmiennych numerycznych: **+**, **-**, ***I***, *****, …

Pozycje kartezjańskie, punkty typu **pointRX** (zawarte w układzie współrzędnym globalnym « **World** » lub zagnieżdżone w lokalnym układzie współrzędnym)

- X, Y, Z, Rx, Ry, Rz

Definicja narzędzi, **tool**

GLOBAL DATA – c.d.



Zmienne analogowe Wejścia/Wyjścia

Zmienne boolowskie (true / false)

- bInit=true

Zmienne cyfrowe Wejścia/Wyjścia, **Di**, **Do**

Lokalne układy współrzędnych, **fFrame**

- X, Y, Z, Rx, Ry, Rz

Zmienne « **Serial** » wymieniane po protokole RS232/422, TCP/IP socket

Transformacje kartezjańskie, **trsf**

Wszystkie zmienne mogą być tablicą jedno lub wielo wymiarową: zdefiniowana liczba elementów, które można wstawiać lub usunąć

- Np.: traj[1], length[3], …

MOTION DESCRIPTOR MDESC

newProject1-mNomSpeed Data

Index	Accel	Vel	Decel	TVel	RVel	Blend	Leave	Reach
0	100	100	100	99 999	99 999	joint	50	50
	%	%	%	mm/s	deg/s		mm	mm

- **vel**, **accel**, **decel**: wartości podawane w procentach %
- **Tvel**: maksymalna prędkość TCP w mm/s
- **Rvel**: maksymalna prędkość TCP w stopniach/s
- Jeżeli nie ma takiej konieczności zaleca się pozostawienie wartości domyślnej = 99999

Z pośród 3 parametrów prędkości, wykorzystany jest ten **najmniejszy**

MOTION DESCRIPTOR MDESC - BLENDING

newProject1-mNomSpeed Data

Index	Accel	Vel	Decel	TVel	RVel	Blend	Leave	Reach
0	100	100	100	99 999	99 999	joint	50	50
	%	%	%	mm/s	deg/s		mm	mm

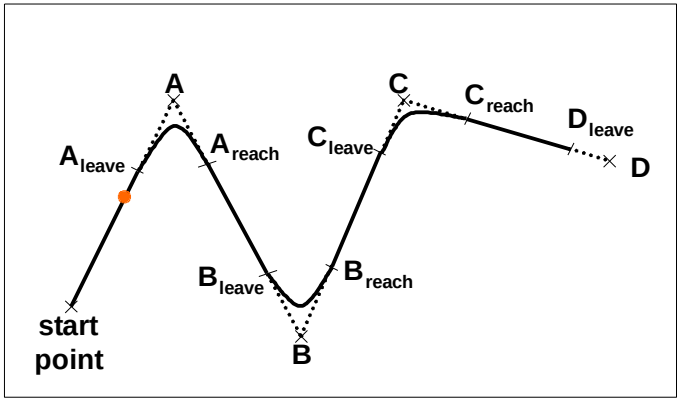
Wartości wyrażone w % odnoszą się do prędkości nominalnych poszczególnych osi

Blend: wygładzanie trajektorii:

- **Joint**: wygładzanie trajektorii realizowane przez ruch JOINT
- **Cartesian**: Punkt TCP narzędzia zachowa ruch na płaszczyźnie (ruch po łuku)
- **Off**: wygładzanie wyłączone, punkt TCP zatrzyma się w punkcie

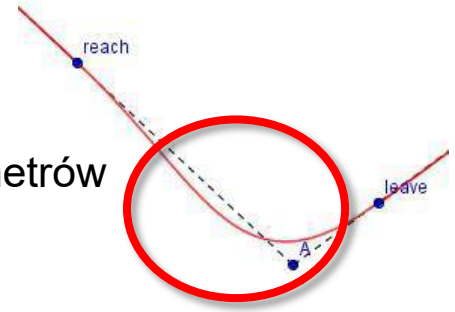
Basic trajectory, without blending

with blending



TCP: **T**ool **C**enter **P**oint

Duże różnice wartości parametrów LEAVE i REACH



WYGŁADZANIE RUCHU: BLENDING

- blending = joint

Robot będzie jechał ruchem jointowym, nie musi być zachowana płaszczyzna podczas przejazdu (ruch najszybszy, najłatwiejszy i najprostszy dla robota a zarazem nieprzewidywalny dla nas).

- blending = cartesian

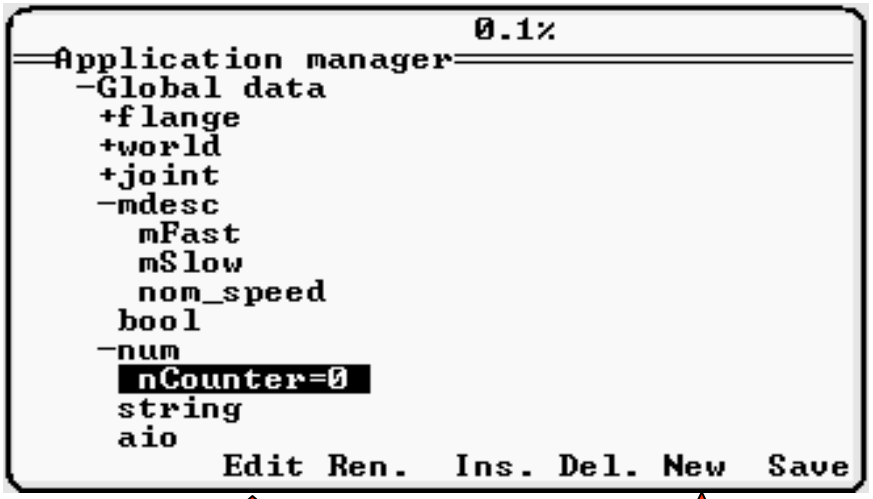
Robot przejeżdżając zachowa płaszczyznę przechodzącą przez 3 punkty (leave, reach i punkt blendingowany) – podobnie jak do kontrolowanego ruchu po łuku.

CZĘŚĆ 1 – Operator CS9

EDYCJA PROGRAMU ZA POMOCĄ MCP

EDYTOR

- Z poziomu menu Application Manager, ustaw kursor na wybranym programie i naciśnij ENTER  aby edytować program lub wybierz opcję EDIT

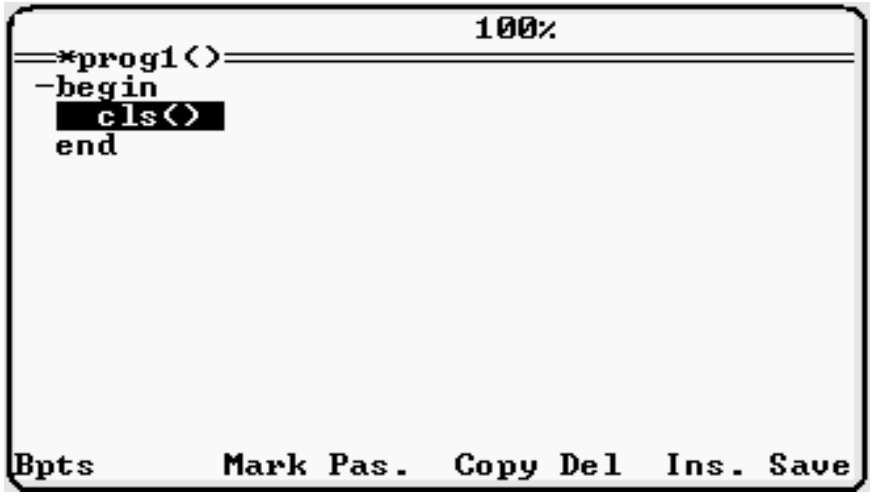


Edycja wybranego programu

Utworzenie nowego programu

EDYCJA PROGRAMU

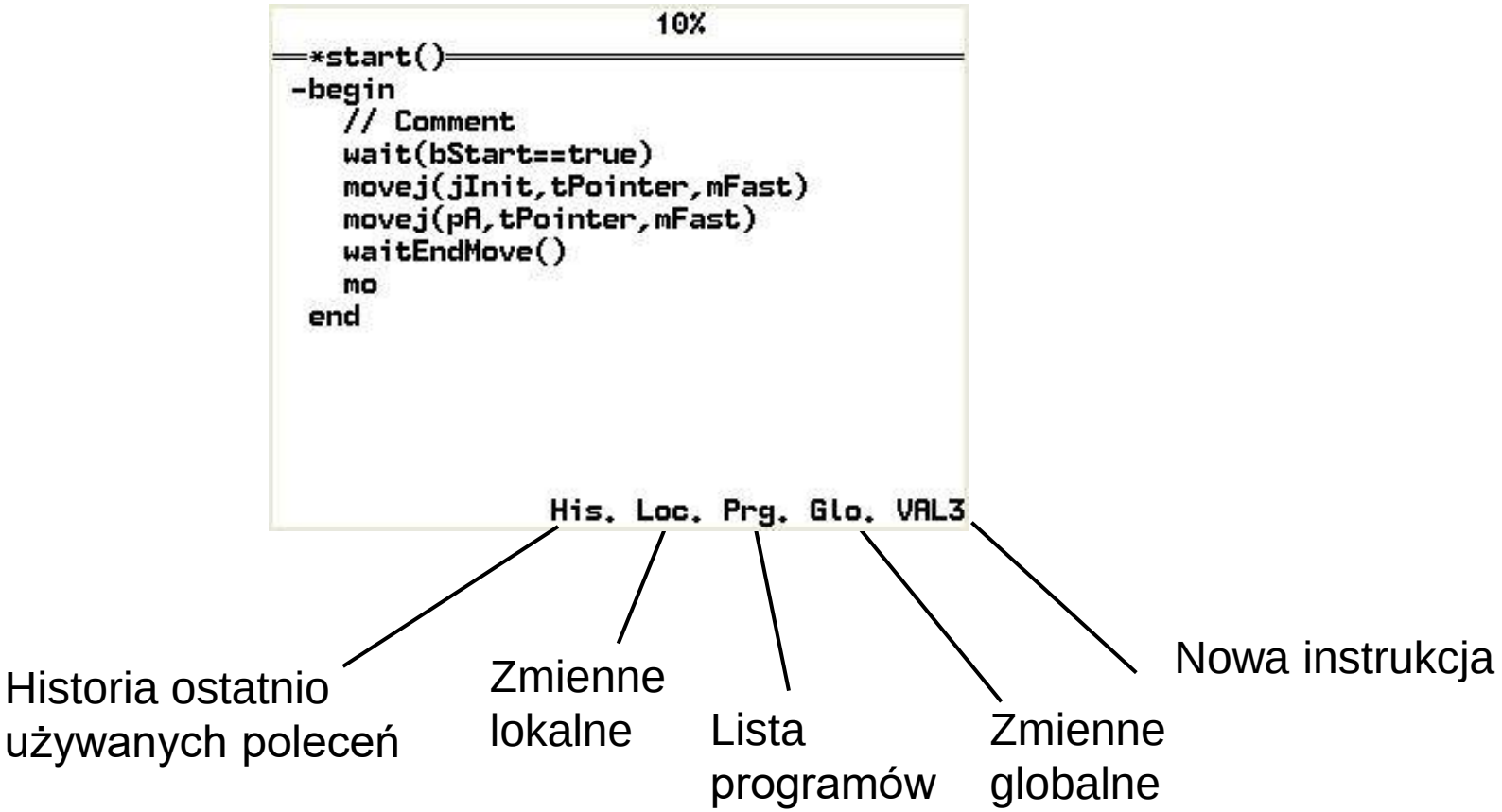
- Składnia musi być poprawna → wszystkie zmienne muszą być utworzone (mogą mieć wartość 0)
- Jedna instrukcja na linię



Wstawia nową linię pod kursorem

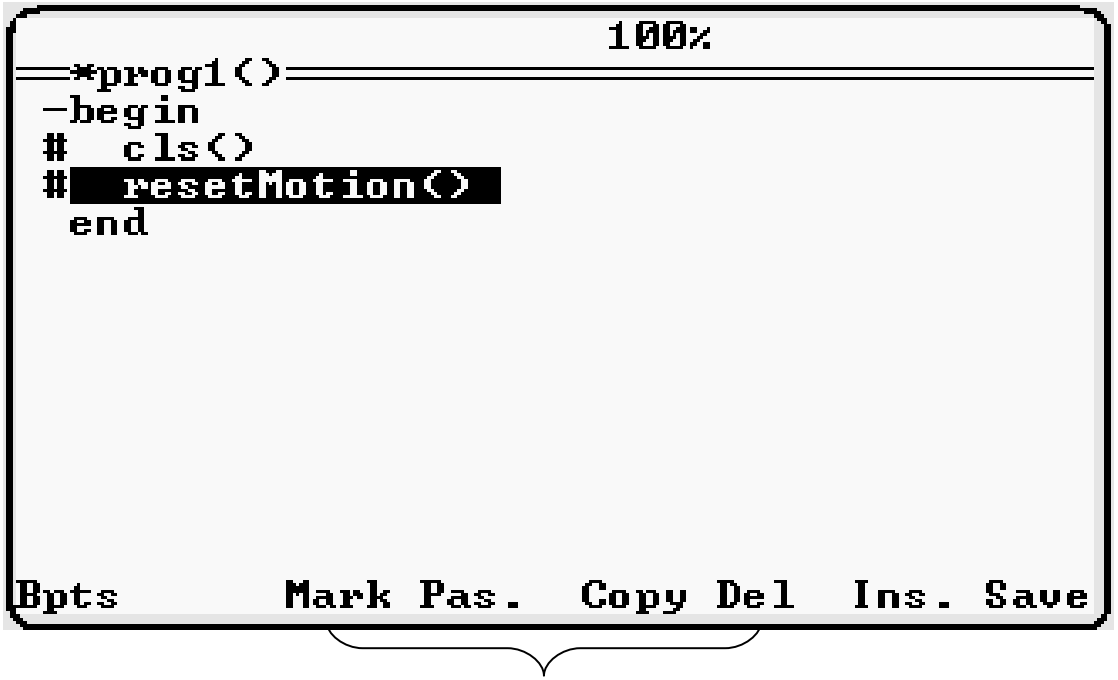
EDYCJA LINII PROGRAMU

- Automatyczne uzupełnianie słów kluczowych: wpisz pierwsze litery polecenia a następnie przycisk VAL3 aby wybrać polecenie z listy
- Naciśnij ENTER aby zakończyć edycję linii programu



OPCJE EDYTORA

- Kopiowanie/Wstawianie linii programu



Zaznaczone linie

Zaznaczanie / Kopiowanie / Kasowanie / Wstawianie

Przesuwanie kliku linii: Zaznacz + Kopiuj + Usuń + Wklej

PODSUMOWANIE

Aby anulować

- Wciśnij « **ENTER** » aby edytować linię programu
- Wciśnij « **ESC** » aby anulować edycje
- Możliwość wprowadzania poleceń znak po znaku
- Możliwość wstawiania poleceń z listy
- Szybkie wstawianie: wybierz menu VAL, naciśnij pierwsze litery polecenia wybierz polecenie i naciśnij « ENTER »

- Przykład: wstawianie instrukcji: **movej(joint, tool,mdesc)**
  - « Ins. » - wstawia nową linię
  - « VAL3 » - wyświetla listę poleceń
  - Wciśnij « m » + « o » + przycisk « Ok » aby dodać instrukcje
  - Uzupełnij 3 parametry za pomocą menu zmiennych globalnych « Glo. »

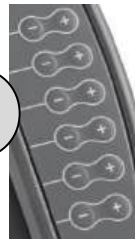
PROGRAMOWANIE RUCHU ROBOTA – ćwiczenie

Ćwiczenie nr 3: Rysowanie trajektorii (MCP) – zadanie 1

Proszę napisać program do wykonywania zadanej trajektorii w zadaniu 1, wykorzystując do tego celu Teach Pendant (pilot):

1. Proszę utworzyć aplikację.
2. W aplikacji proszę utworzyć narzędzie o odpowiednich wymiarach (zmienna tool). Następnie sprawdzić czy są one poprawne poruszając robotem – koniec narzędzia powinien pozostawać nieruchomy podczas ruchów RX, RY, RZ.
3. Proszę utworzyć zmienne punktów oraz nauczyć ich pozycje.
4. Proszę wyedytować program START() i wstawić do niego instrukcje ruchu.
5. Na końcu programu proszę wpisać instrukcję: WaitEndMove().

CONNECTION MOVE



2

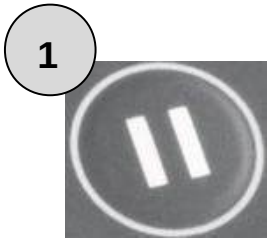
Ruch w trybie ręcznym za pomocą teachpendanta lub poprzez zwolnienie hamulców …

3

Naciśnięcie przycisku Move/hold (i przytrzymanie) w dowolnym trybie pracy: ruch z małą prędkością do punktu przerwania. Robot zatrzyma się w punkcie przerwania

W tym momencie, można zmienić tryb pracy i ręcznie poruszać robotem, na przykład celem ominięcia przeszkody.

Uwaga na przeszkody



1

Przerwanie cyklu pracy

Punkt przerwania

Trajektoria

Punkt pobierania

Punkt odkładania

4

Naciśnięcie przycisku Move/hold (i przytrzymanie, w trybie ręcznym) lub pojedyncze naciśnięcie w trybie automatycznym: wznawia wykonywanie cyklu pracy

Wszystkie czynności bez zatrzymywania aplikacji!

KONTYNUACJA RUCHU – ćwiczenie

Ćwiczenie nr 4: Kontynuacja ruchu

Proszę wykonać program z ćwiczenia nr 3 (zadanie 1) w trybie automatycznym z małą prędkością i przerwać jego wykonanie. Następnie wznowić pracę programu z tego samego miejsca.

Proszę wykonać program z ćwiczenia nr 3 (zadanie 1) w trybie automatycznym z małą prędkością i przerwać jego wykonanie. Następnie zmienić pozycję robota i wznowić wykonywanie programu.

SZKOLENIE Z OBSŁUGI ROBOTÓW STÄUBLI

cz.2 – poziom user

Seria robotów: TX2/**CS9**

TABELA ZAGADNIEŃ – USER

- **92** – SRS – STÄUBLI ROBOTICS SUITE
- **127** – ĆWICZENIE 5 (zadanie nr 2 – „*Blending*”)
- **127** – OPTIMIZE LAB – RECORDER
- **138** – ĆWICZENIE 6 (*Nagrywanie trajektorii pracy robota*)
- **139** – TASK MANAGER
- **143** – ĆWIECZENIE NR 7 (*task manager i debugging*)
- **144** – OBSŁUGA I/O
- **159** – OBLICZANIE POZYCJI DOJAZDU DO PUNKTÓW KARTEZJAŃSKICH
- **165** – ĆWICZENIE NR 8 (zadanie nr 3 – „*Zamiana miejsc*”)
- **166** – UKŁADY WSPÓŁRZĘDNYCH
- **175** – ĆWICZENIE NR 9 (*Nauka układów współrzędnych*)

TX-TS-RX-RS-TP/**CS8**

TX2/**CS9**

CZĘŚĆ 2 – USER CS9

SRS
STÄUBLI ROBOTICS SUITE

SOFTWARE SUITE POD WINDOWS'A

- Idealny do realistycznej symulacji, programowania w trybie off-line i zdalnego dostępu do oddalonych od programisty robotów

SRC - **S**täubli **R**obotics **C**ontrols

SRS – **S**täubli **R**obotics **S**uite **2019**

SPERSONALIZOWANA PRZESTRZEŃ PRACY

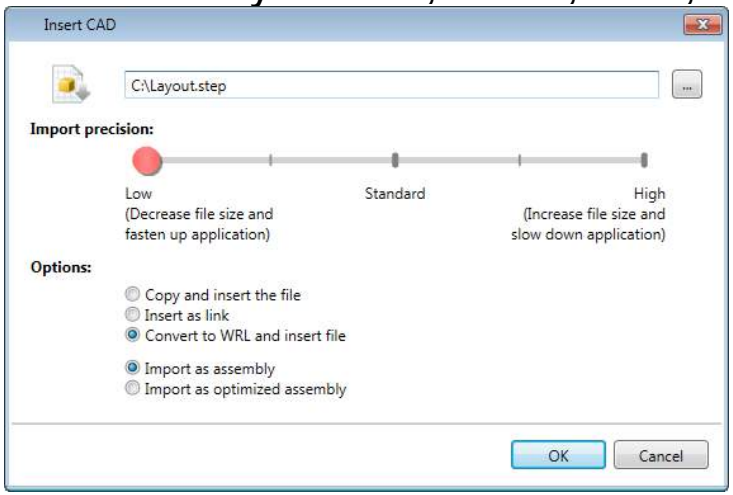
- Bezpłatna możliwość zarządzania kilkoma panelami (kompatybilny soft)

IMPORT PLIKÓW CAD

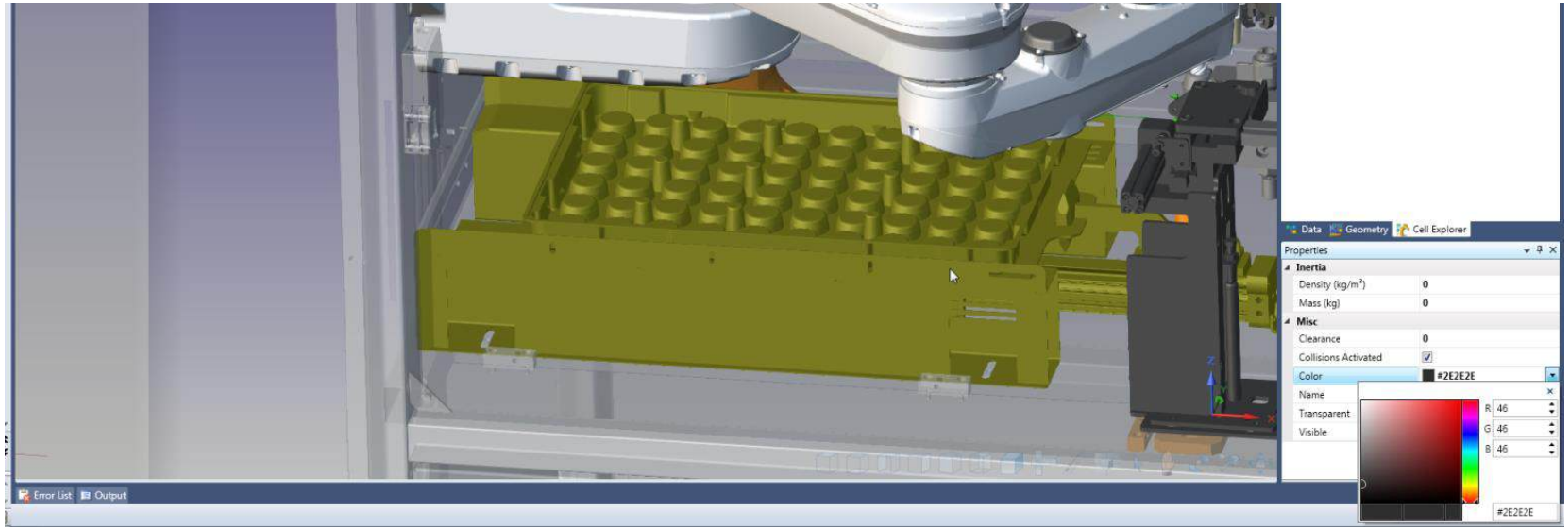
Development Studio

Wymagana licencja do używania tych funkcji

- Import kreator (obsługiwane formaty: **STEP**, **IGES**, **STL**, **WRL**)



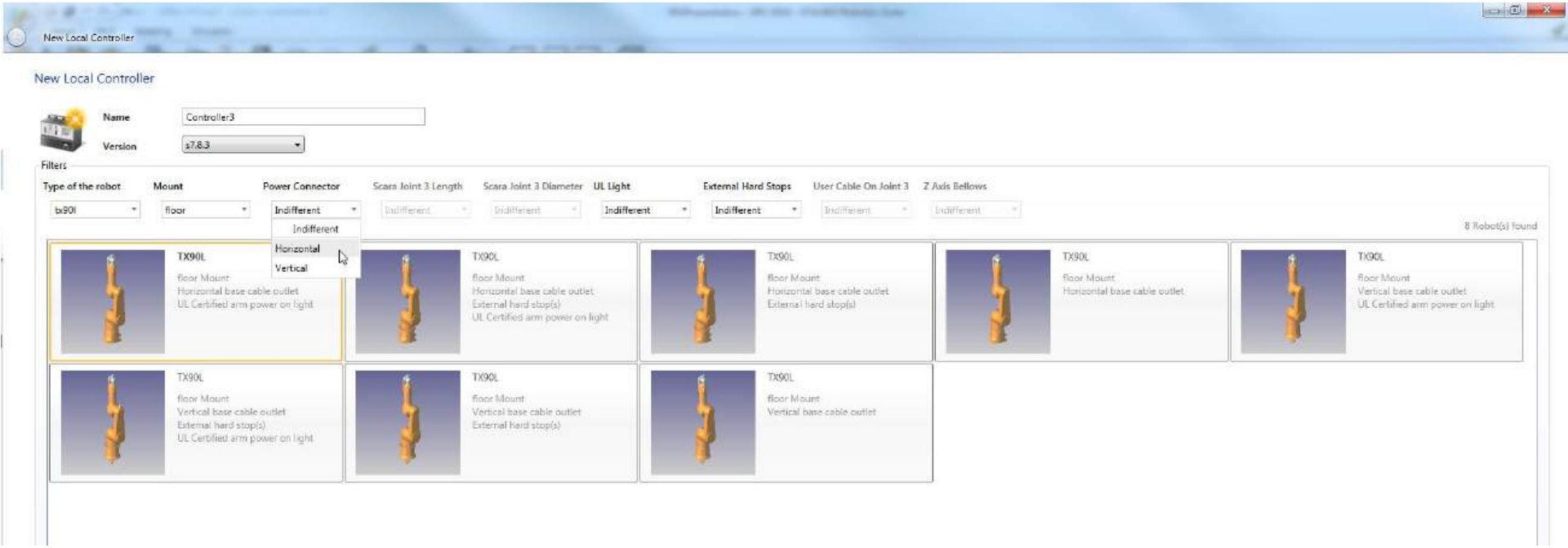
- Modyfikacja właściwości komponentów 3D (pozycja, kolor, itp.)



ASYSTENT WYBORU INSTALOWANEGO ROBOTA W CELI

Funkcja darmowa

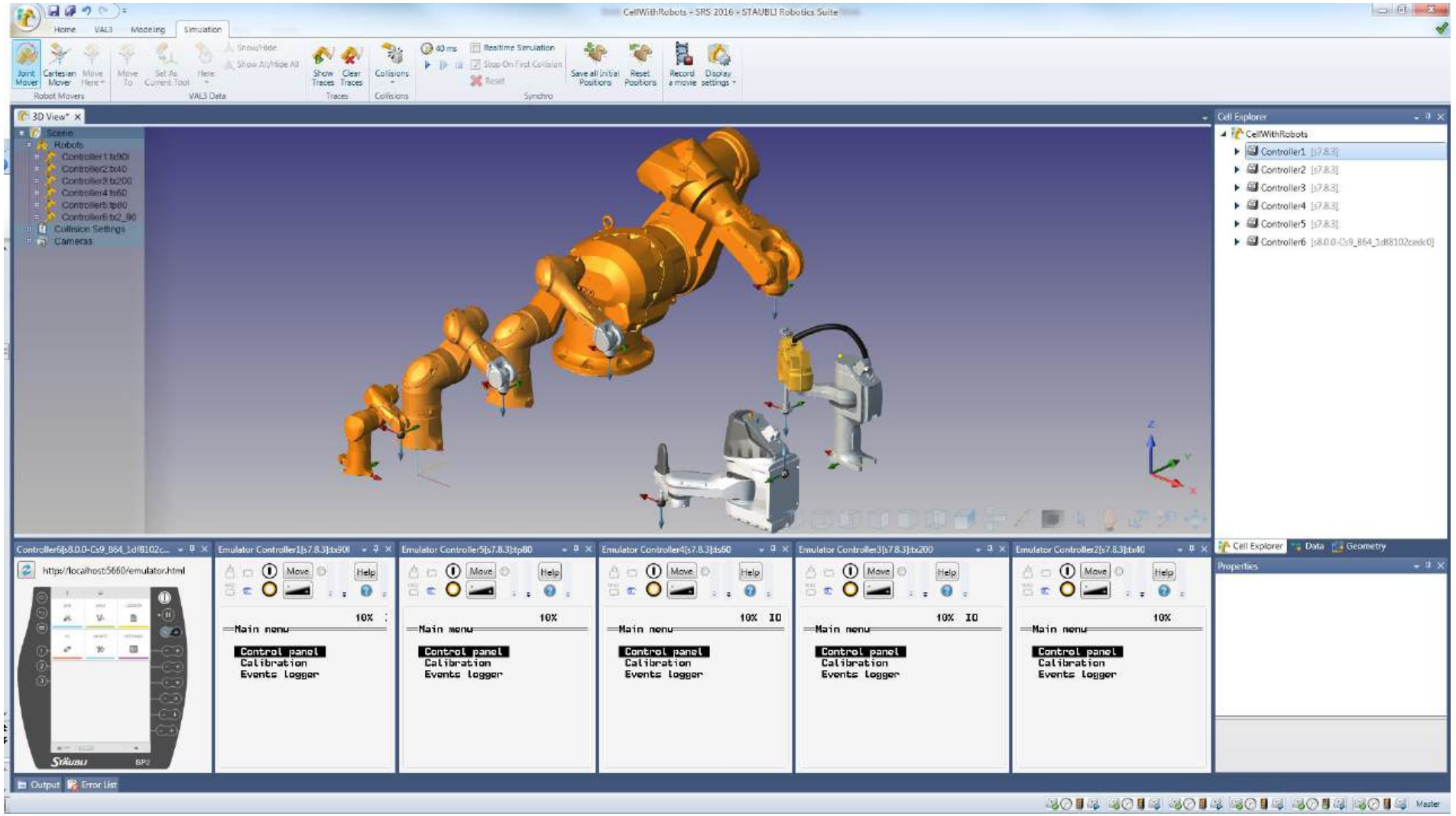
- Wiele wersji modelu ramienia z opcjami (detekcja kolizji jest bardziej precyzyjna: wirtualny robot identyczny z rzeczywistym)



SYMULACJA WIELU ROBOTÓW

Funkcja darmowa

- Po zainstalowaniu w celi roboty są wyświetlane na scenie 3D

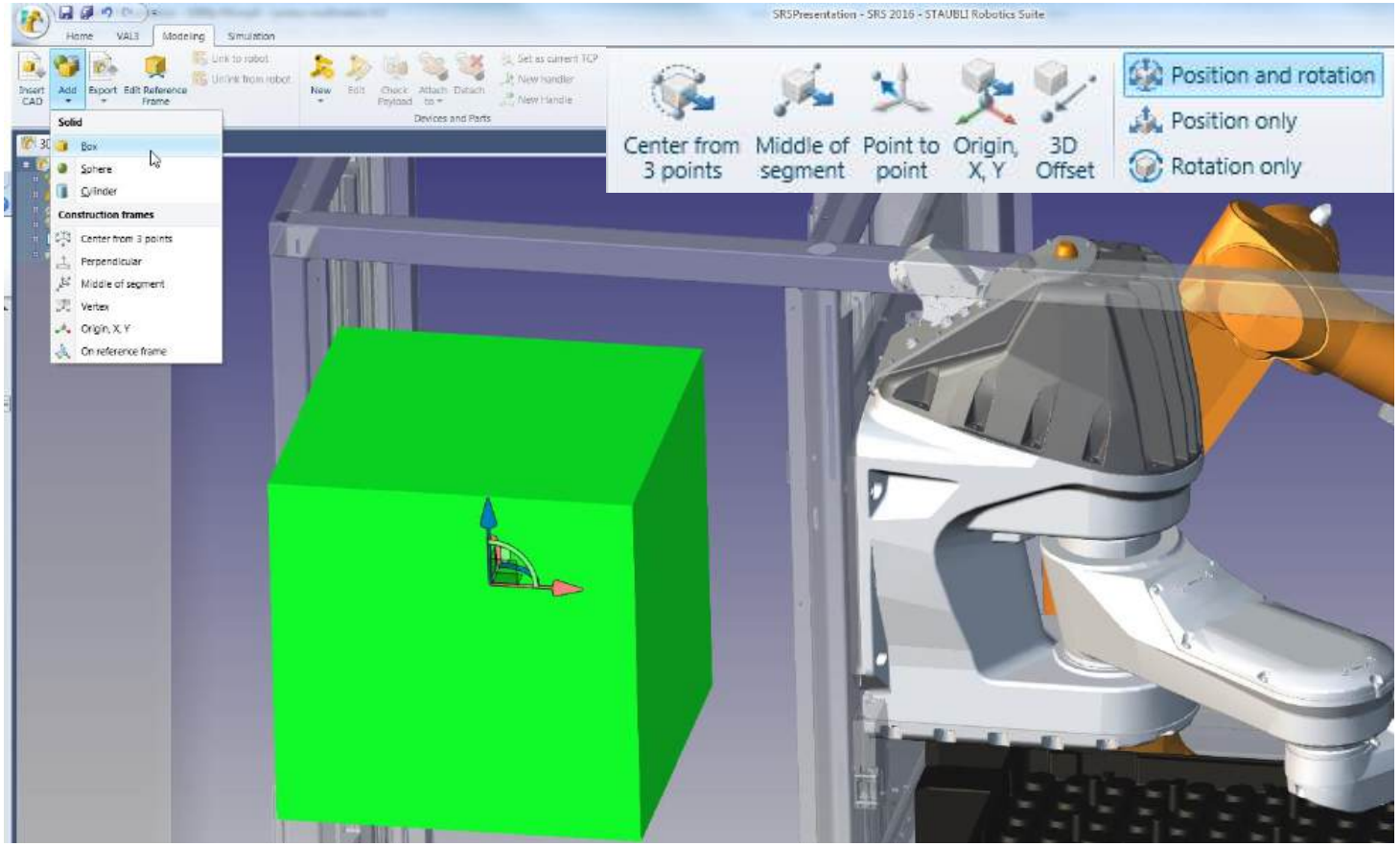


MODELE 3D

Development Studio

Wymagana licencja do używania tych funkcji

- Oparte na formach podstawowych (pudełko, cylinder i kula), idealne do uzupełnienia zaimportowanego modelu bez konieczności ponownej pracy z programem CAD
- Pozycjonowanie obiektów jest łatwe w użyciu i oparte na wykrywaniu wierzchołków

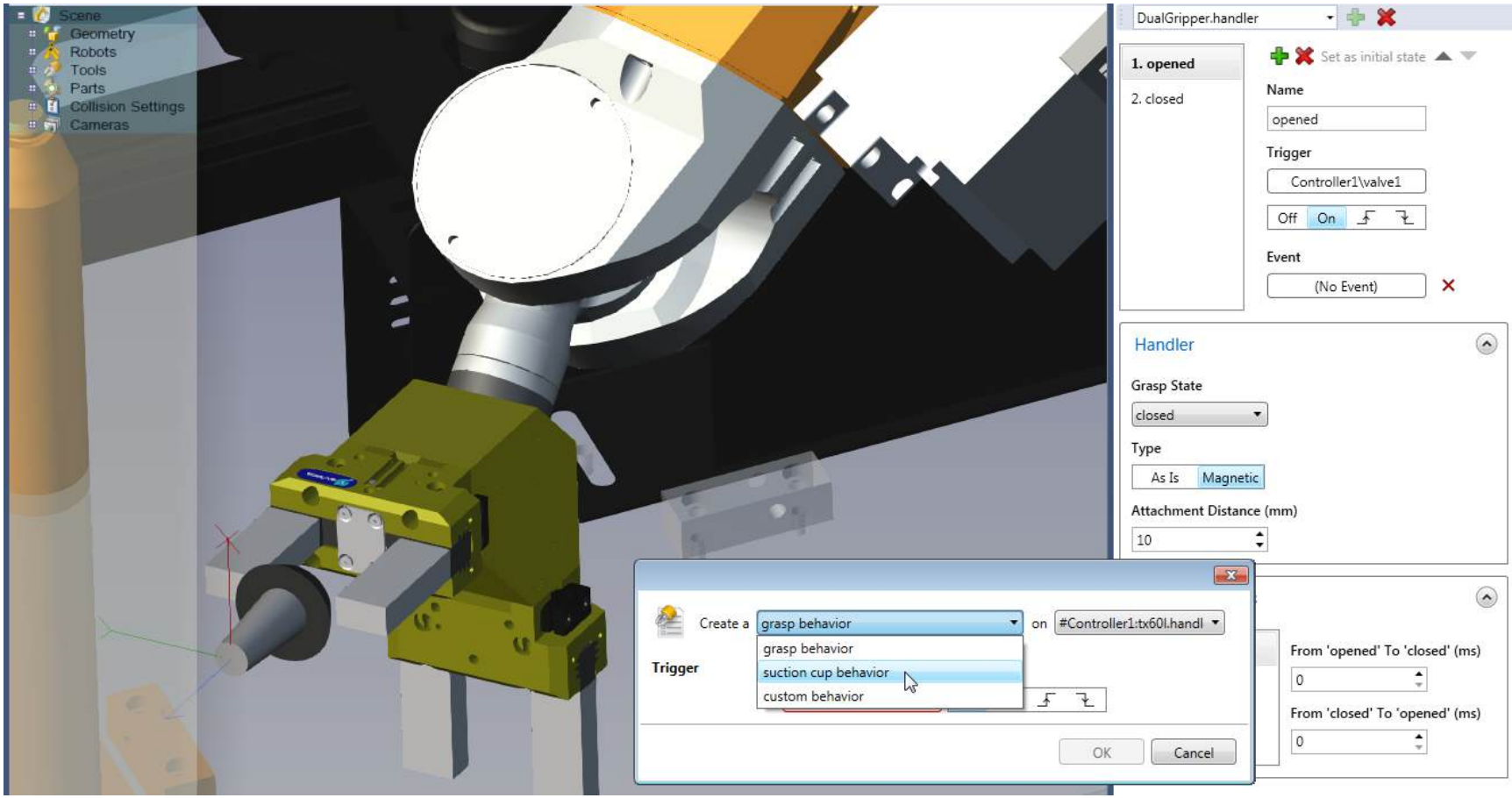


TWORZENIE NARZĘDZI

Development Studio

Wymagana licencja do używania tych funkcji

- Na podstawie zaimportowanych plików CAD lub utworzonych w SRS, łatwa konfiguracja do obsługi części lub innych narzędzi (zmieniarka narzędzi)

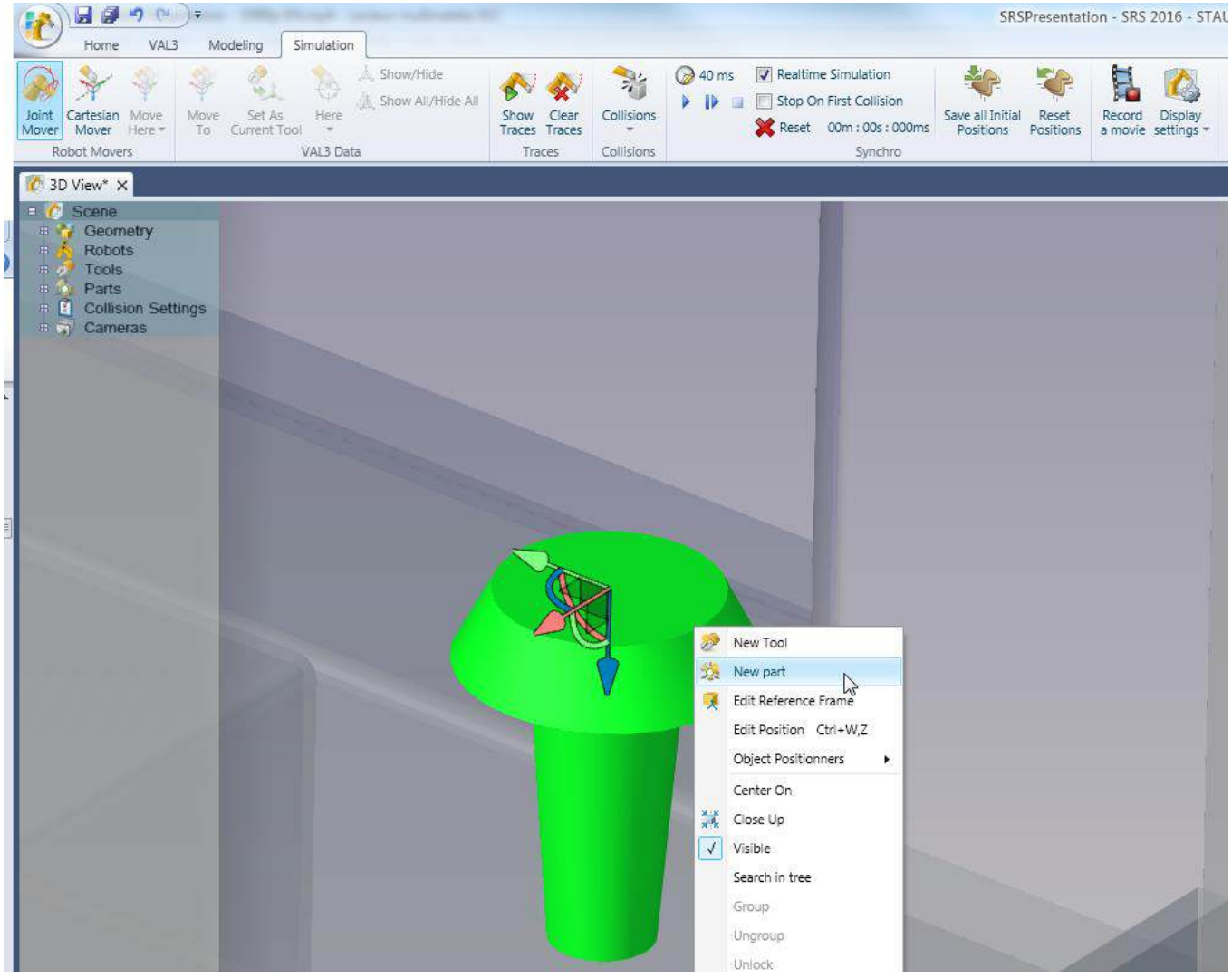


CZĘŚCI RUCHOME

Development Studio

Wymagana licencja do używania tych funkcji

- Na podstawie zaimportowanych plików CAD lub utworzonych w SRS, łatwa konfiguracja do obsługi w kilku pozycjach

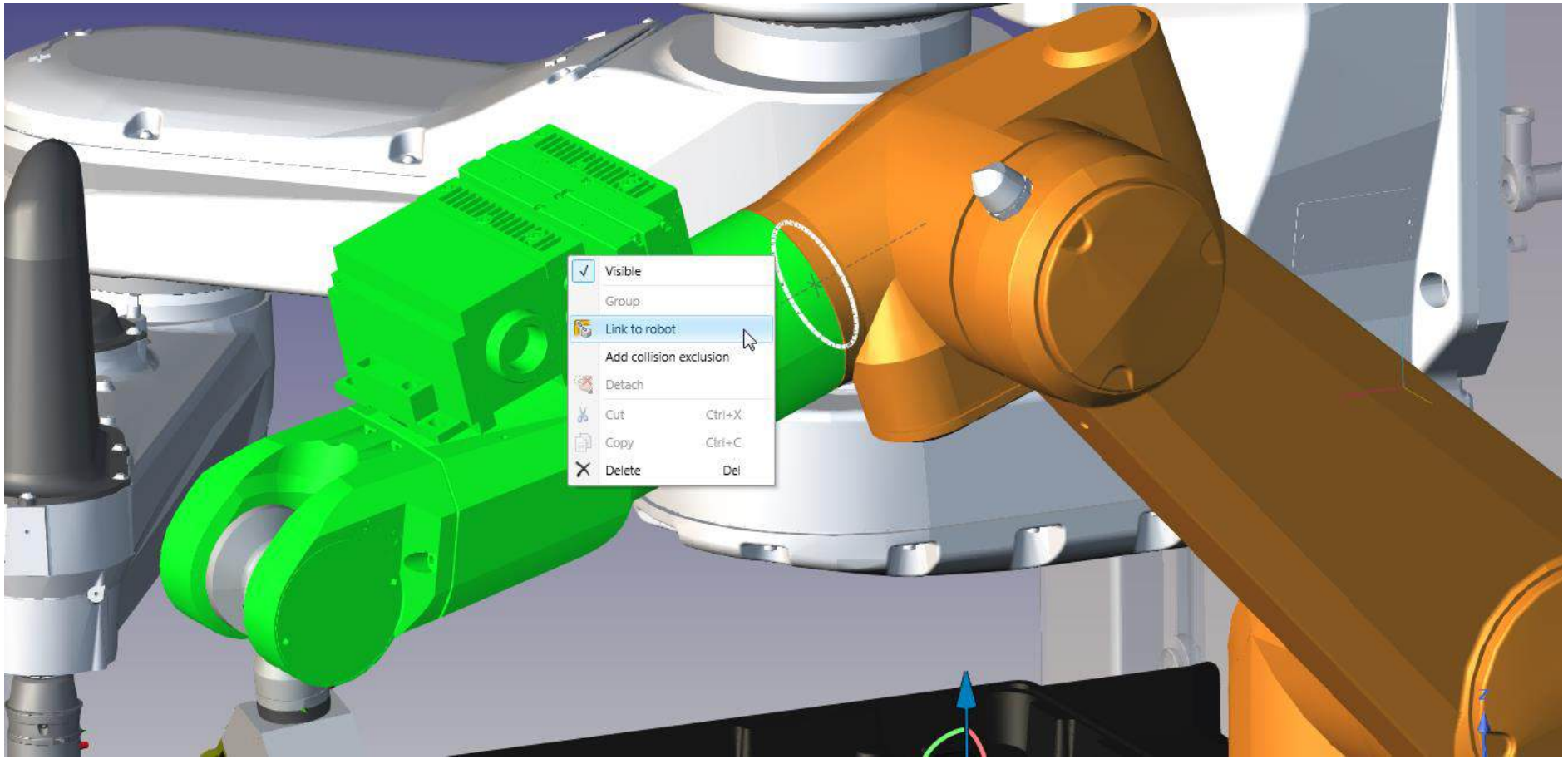


DODATKOWE OBCIĄŻENIE (PAYLOAD)

Development Studio

Wymagana licencja do używania tych funkcji

- Dołączone obciążenie na ramieniu i wykrywanie kolizji
- Automatyczne obliczanie bezwładności

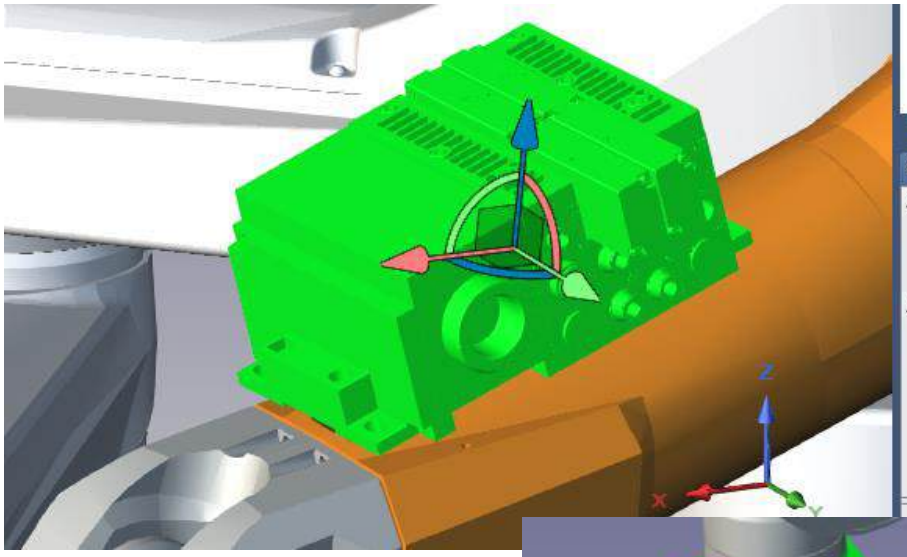


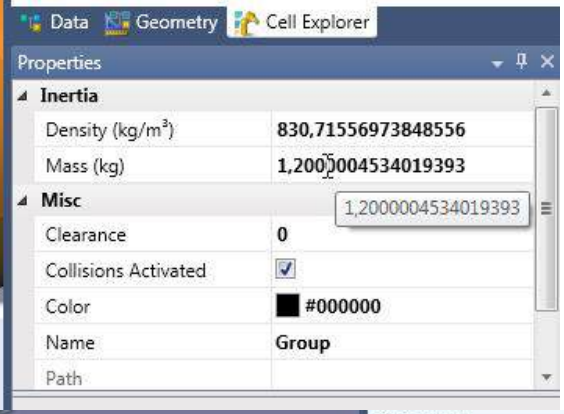
MASY I DEFINICJA BEZWŁADNOŚCI

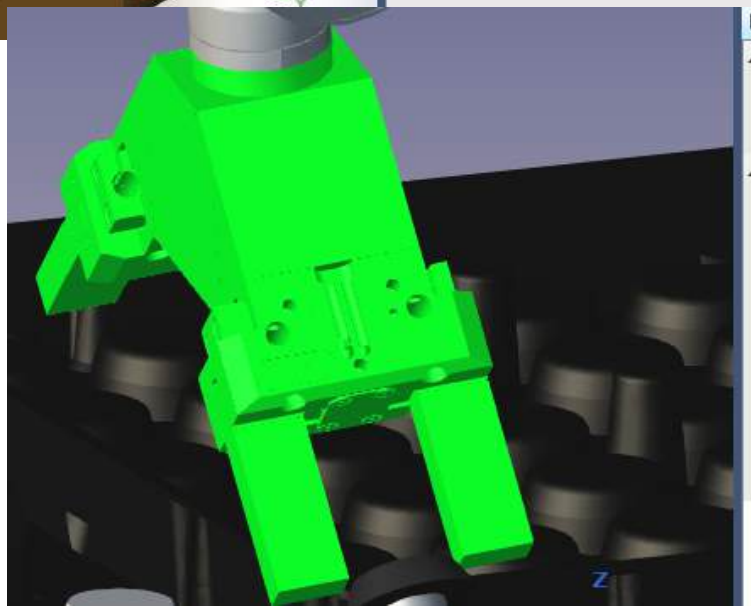
Development Studio

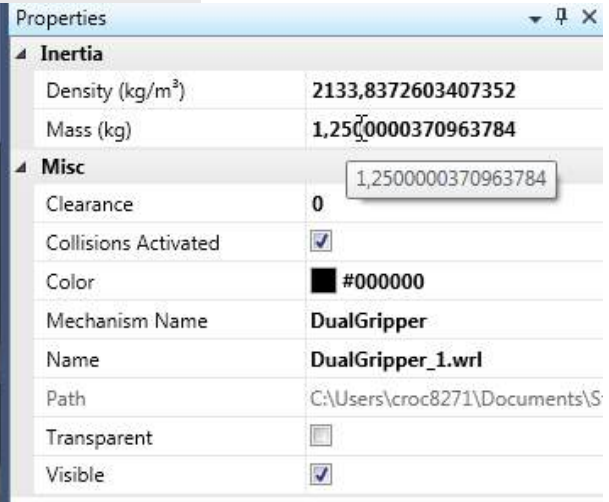
Wymagana licencja do używania tych funkcji

- Określ masę lub gęstość narzędzi, części, dodatkowego ładunku







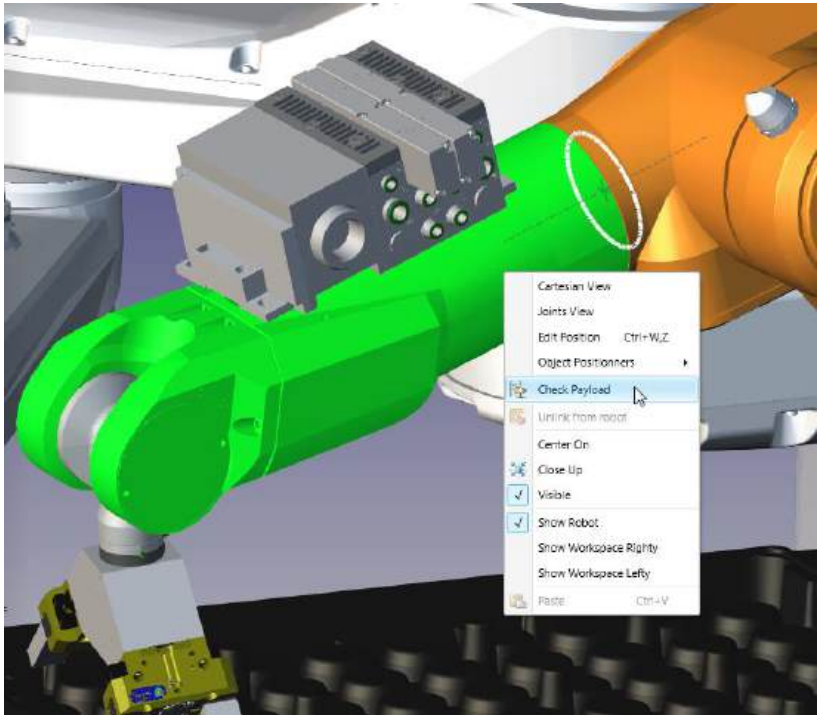


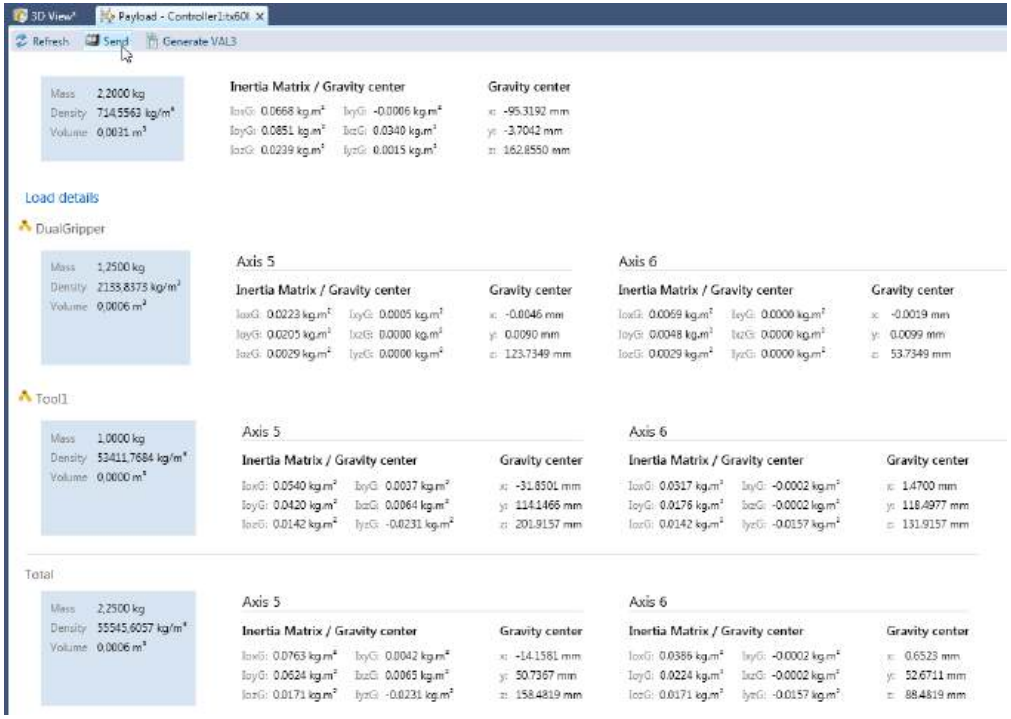
SPRAWDZENIE PARAMETRÓW OBCIĄŻENIA

Development Studio

Wymagana licencja do używania tych funkcji

- Automatyczne obliczanie bezwładności (środek ciężkości, bezwładność)
- Parametry obciążenia są brane pod uwagę podczas realistycznej symulacji



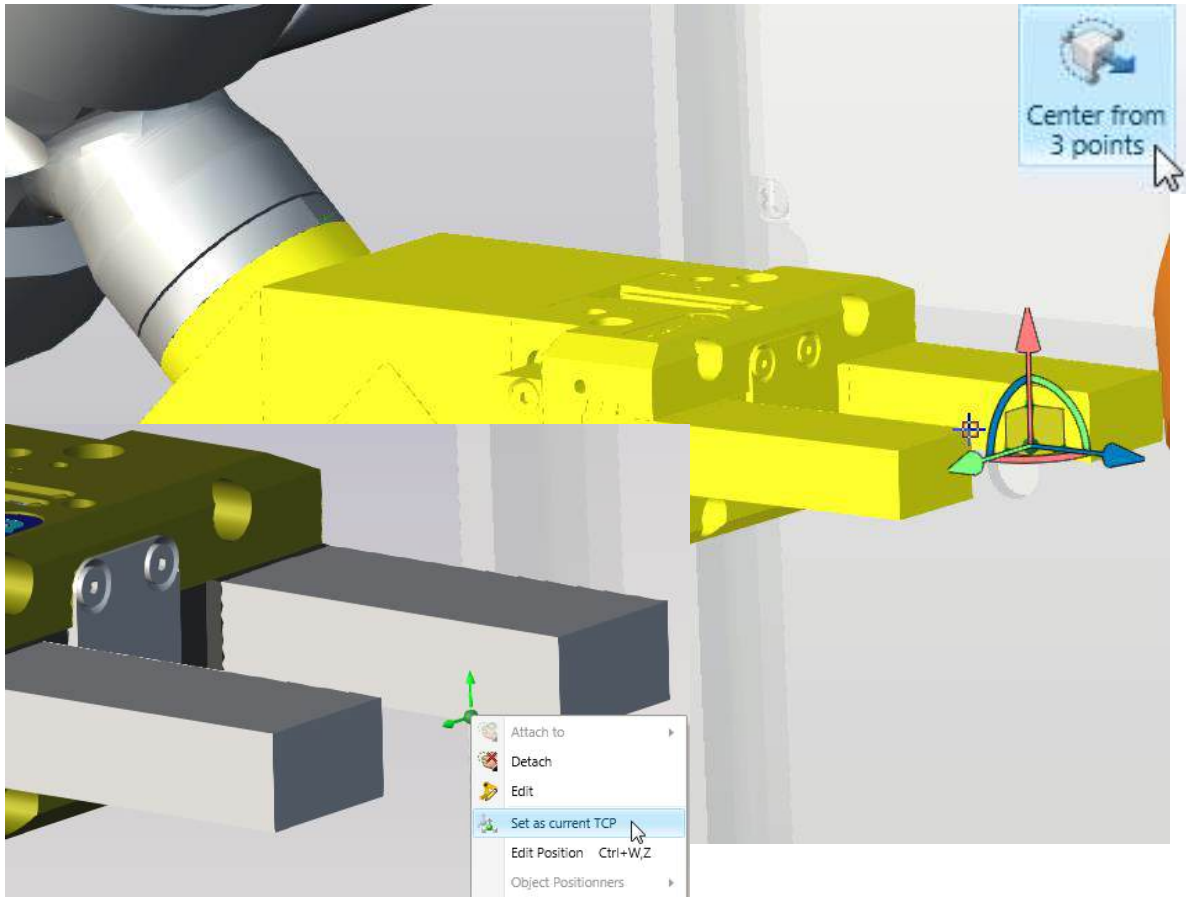


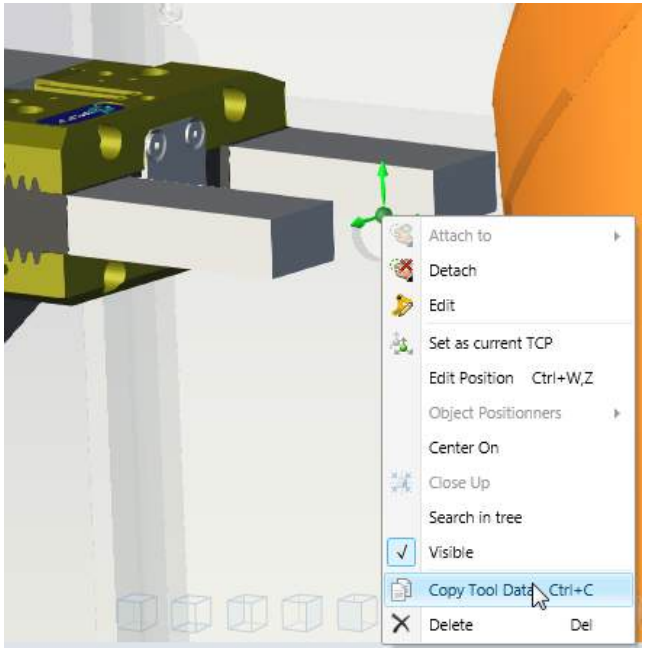
OBLICZENIE TCP – TOOL CENTER POINT, CENTRALNEGO PUNKTU NARZĘDZIA

Development Studio

Wymagana licencja do używania tych funkcji

- Szybkie pozycjonowanie TCP za pomocą graficznych pozycjonerów
- Skopiuj i wklej ze sceny 3D do zmiennej TOOL projektu VAL 3

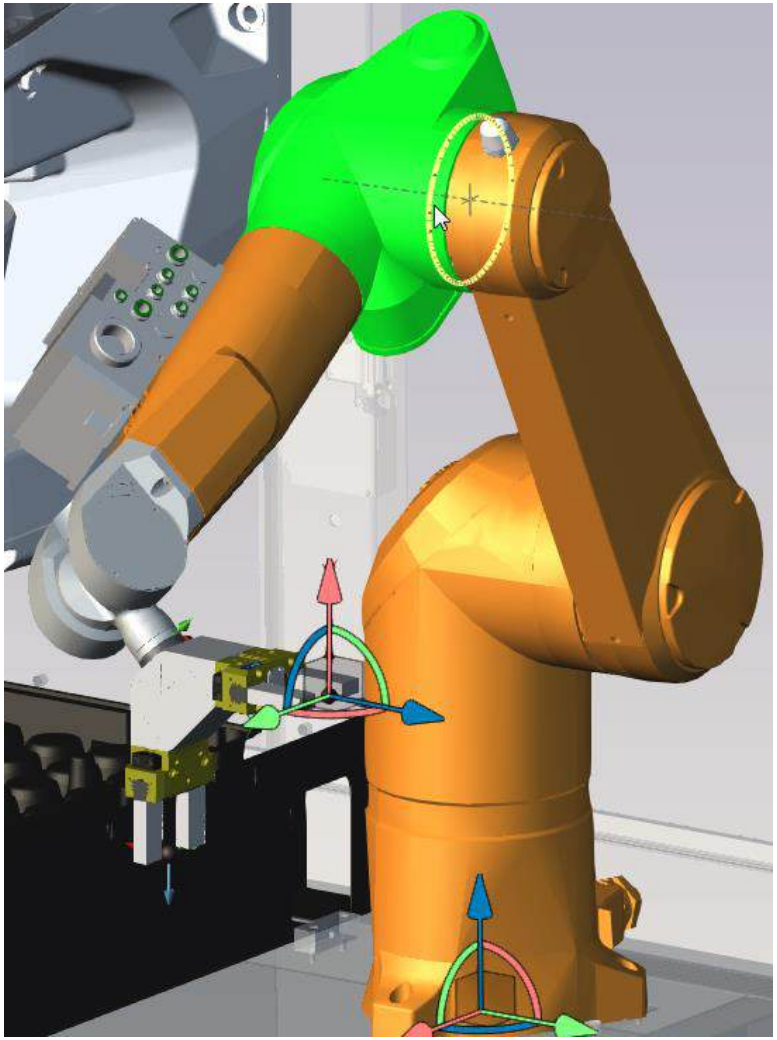


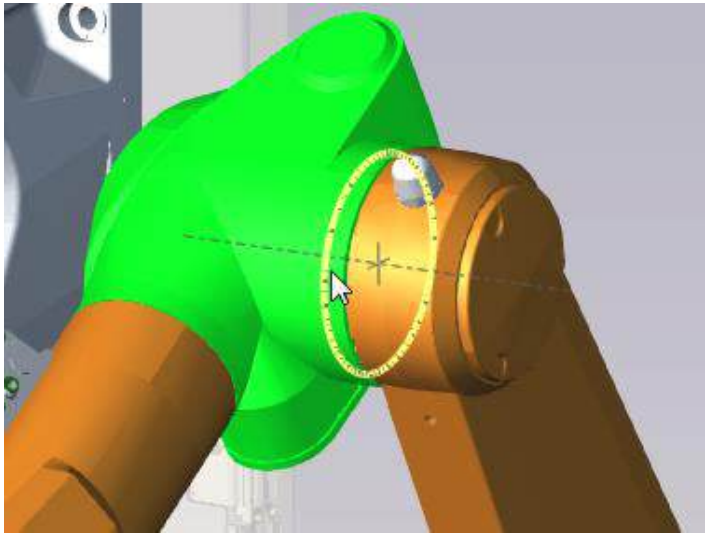


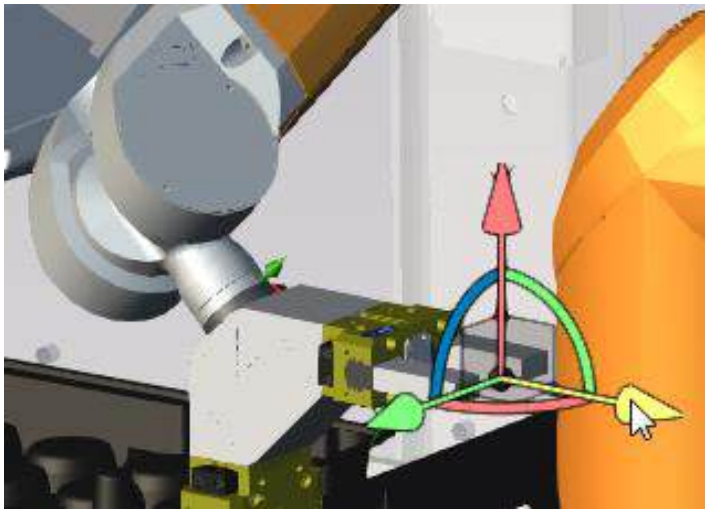
RUCHY RĘCZNE (GRAFIKA)

Funkcja darmowa

- Kliknij i przesuń na osiach TCP (ruchy kartezjańskie) lub na danej osi robota (ruchy kątowe)





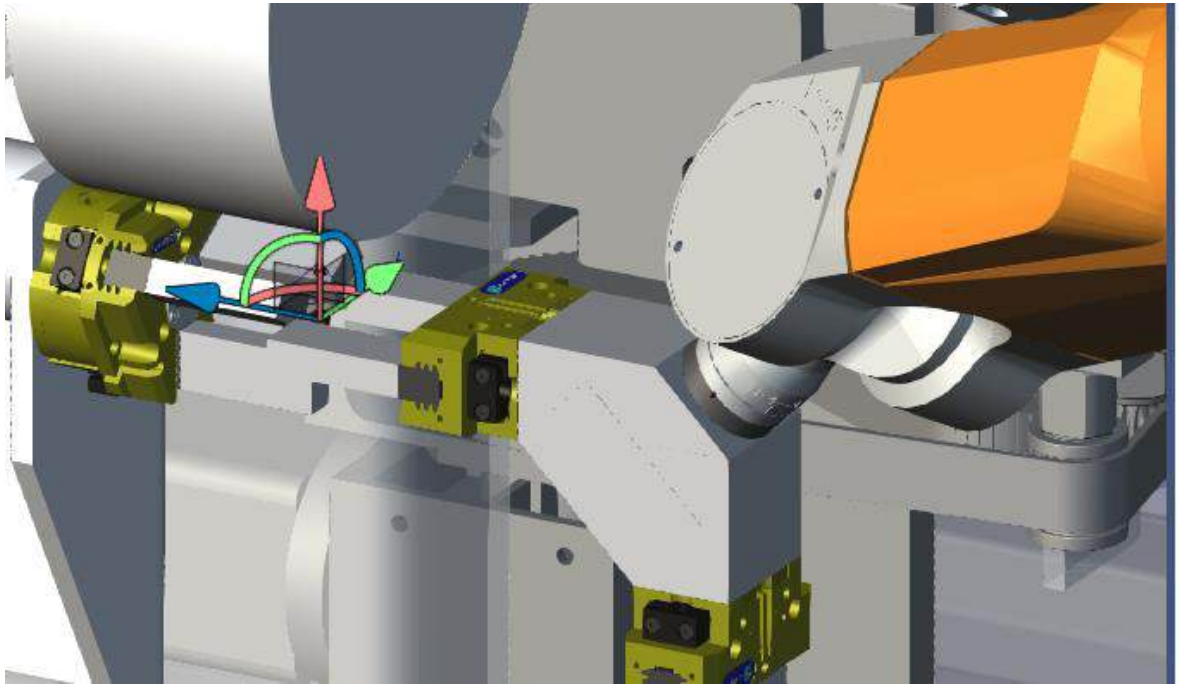


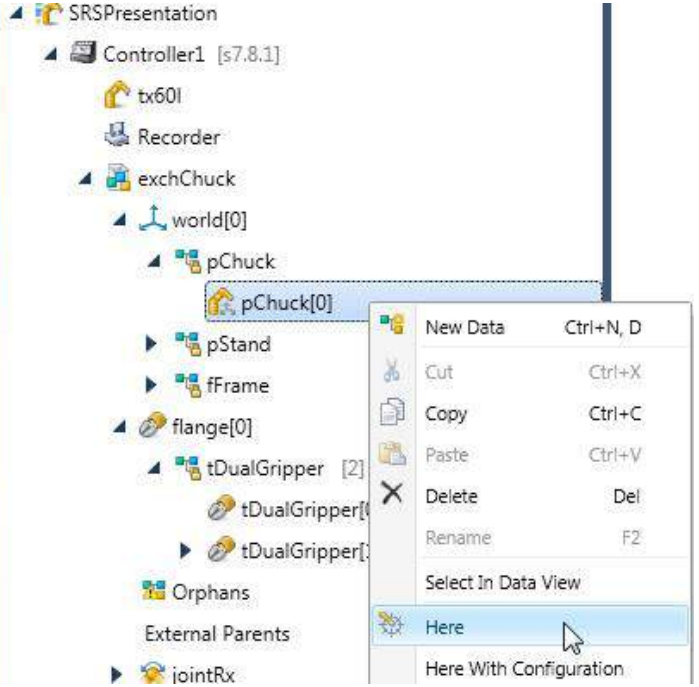
BEZPOŚREDNIE NAUCZANIE PUNKTÓW

Development Studio

Wymagana licencja do używania tych funkcji

- Łatwe pozycjonowanie robota za pomocą ruchu ręcznego lub narzędzi graficznych
- Proste kliknięcie prawym klawiszem VAL3 dla nauczenia



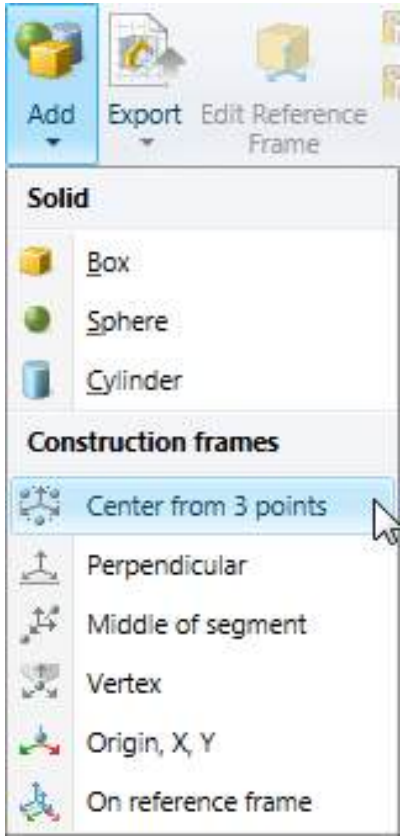


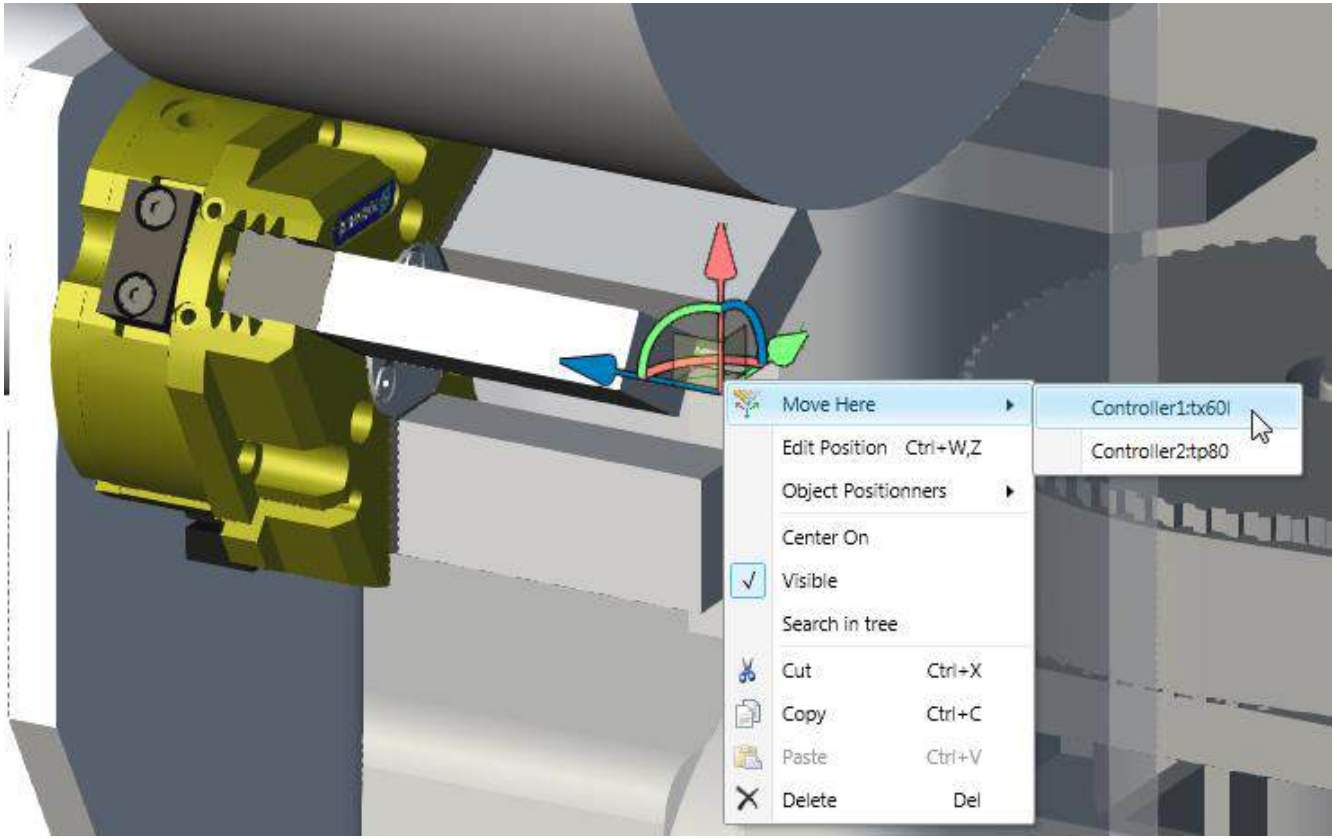
PUNKTY GRAFICZNE

Development Studio

Wymagana licencja do używania tych funkcji

- Punkty można tworzyć graficznie bez znajomości języka VAL
- Może być następnie użyty do zdefiniowania VAL 3 punktów lub układów współrzędnych
- Roboty można przenosić do punktów graficznych w celu analizy zasięgu



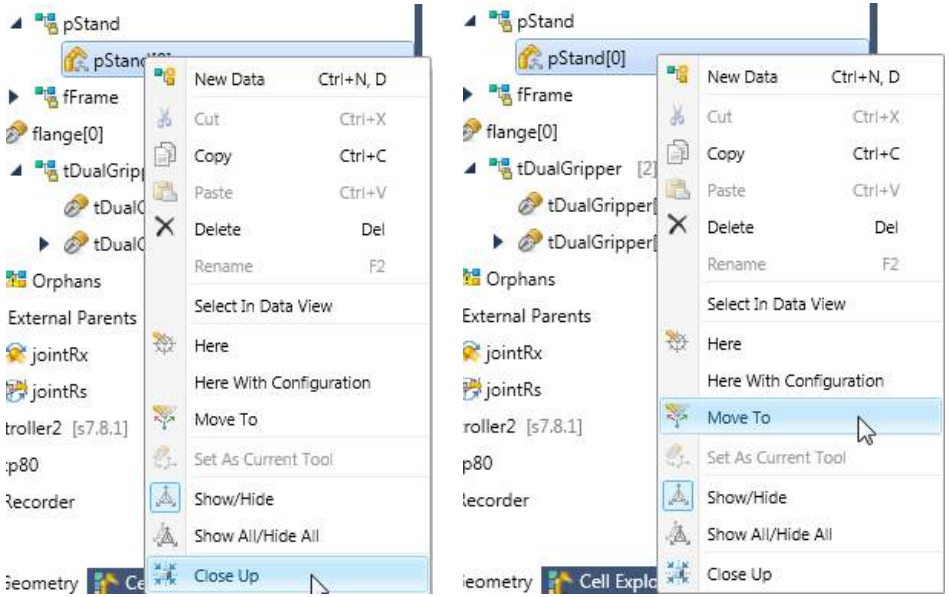


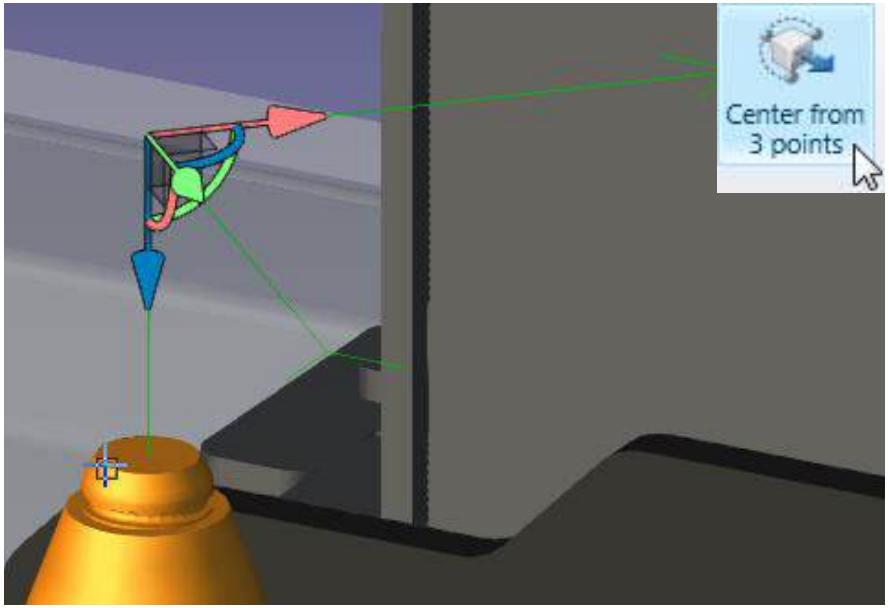
POZYCJA VAL3 – PUNKTY & UKŁADY WSPÓŁRZĘDNYCH

Development Studio

Wymagana licencja do używania tych funkcji

- VAL3 punkty i układy współrzędnych można łatwo zlokalizować w scenie 3D (zbliżenie)
- Ponadto można je ustawić graficznie za pomocą pozycjonera obiektów i sprawdzić zasięg (**Move to**)



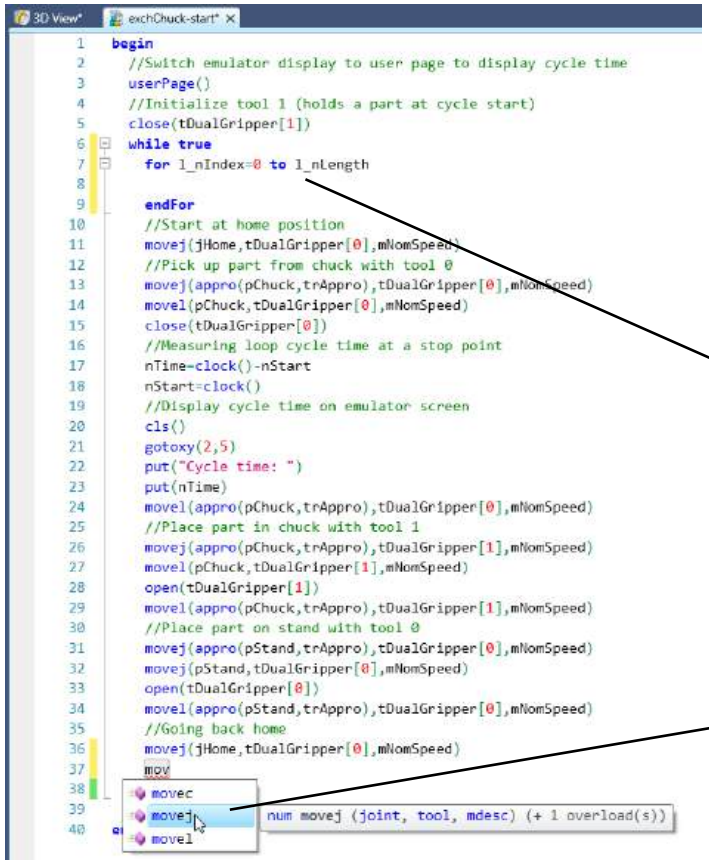


EDYCJA KODU VAL 3

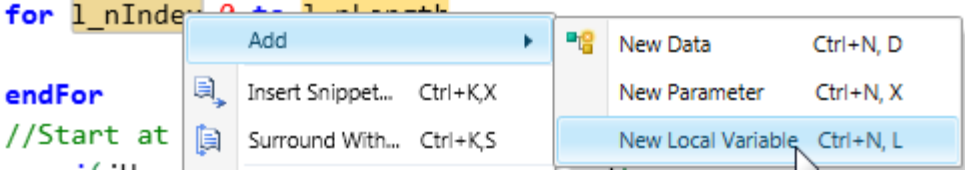
Development Studio

Wymagana licencja do używania tych funkcji

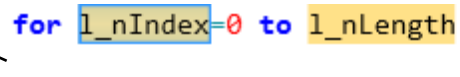
- Intuicyjny edytor ze zintegrowaną obsługą do programowania, kolorowania składni, powiększania i wstawiania fragmentów
- Zmienne można tworzyć bezpośrednio z programu



→ *Tworzenie zmiennych*



→ *Kawałek*



→ *zoom*

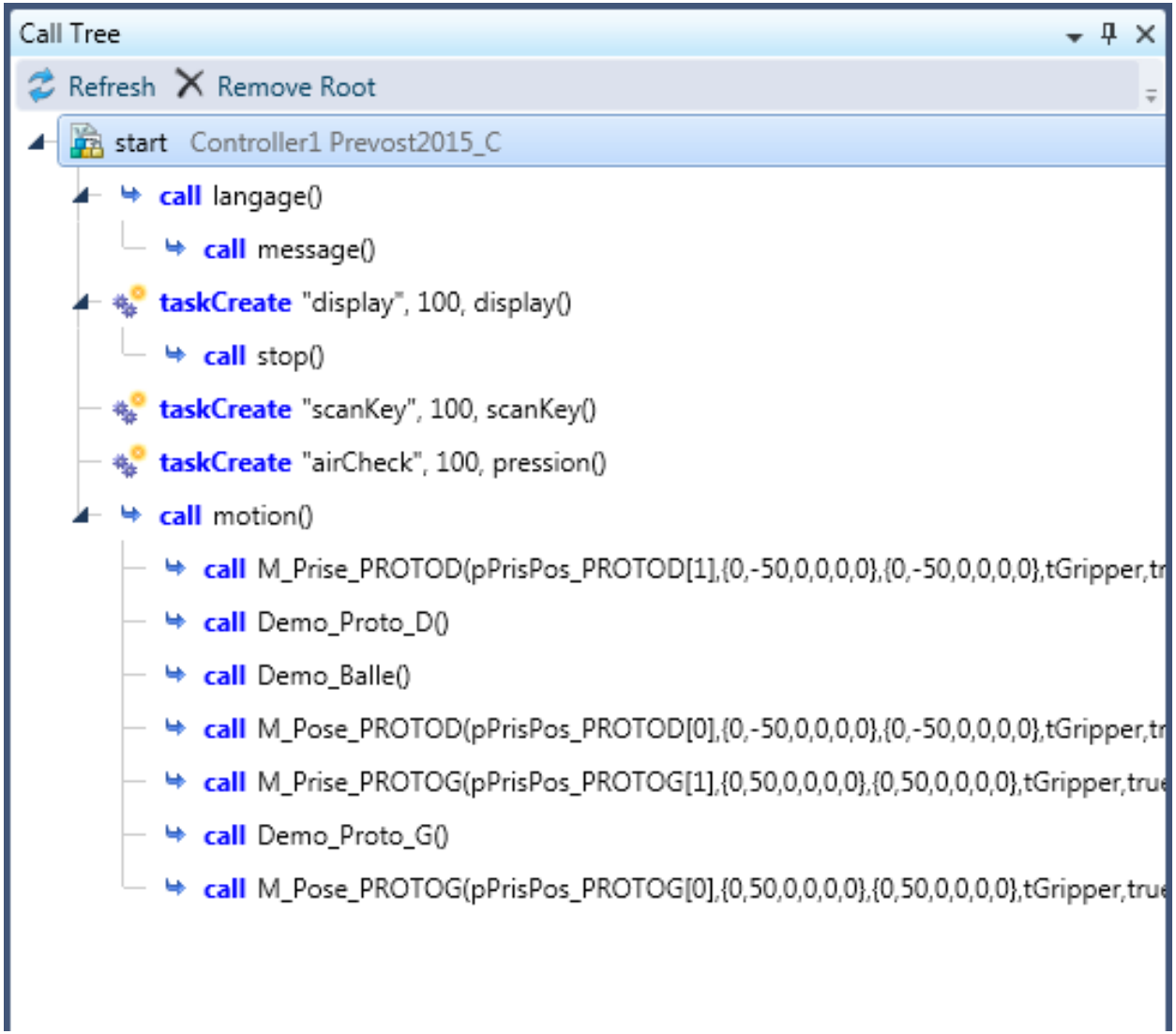
Nowość

→ *Automatyczne uzupełnianie*

STRUKTURA APLIKACJI

Funkcja darmowa

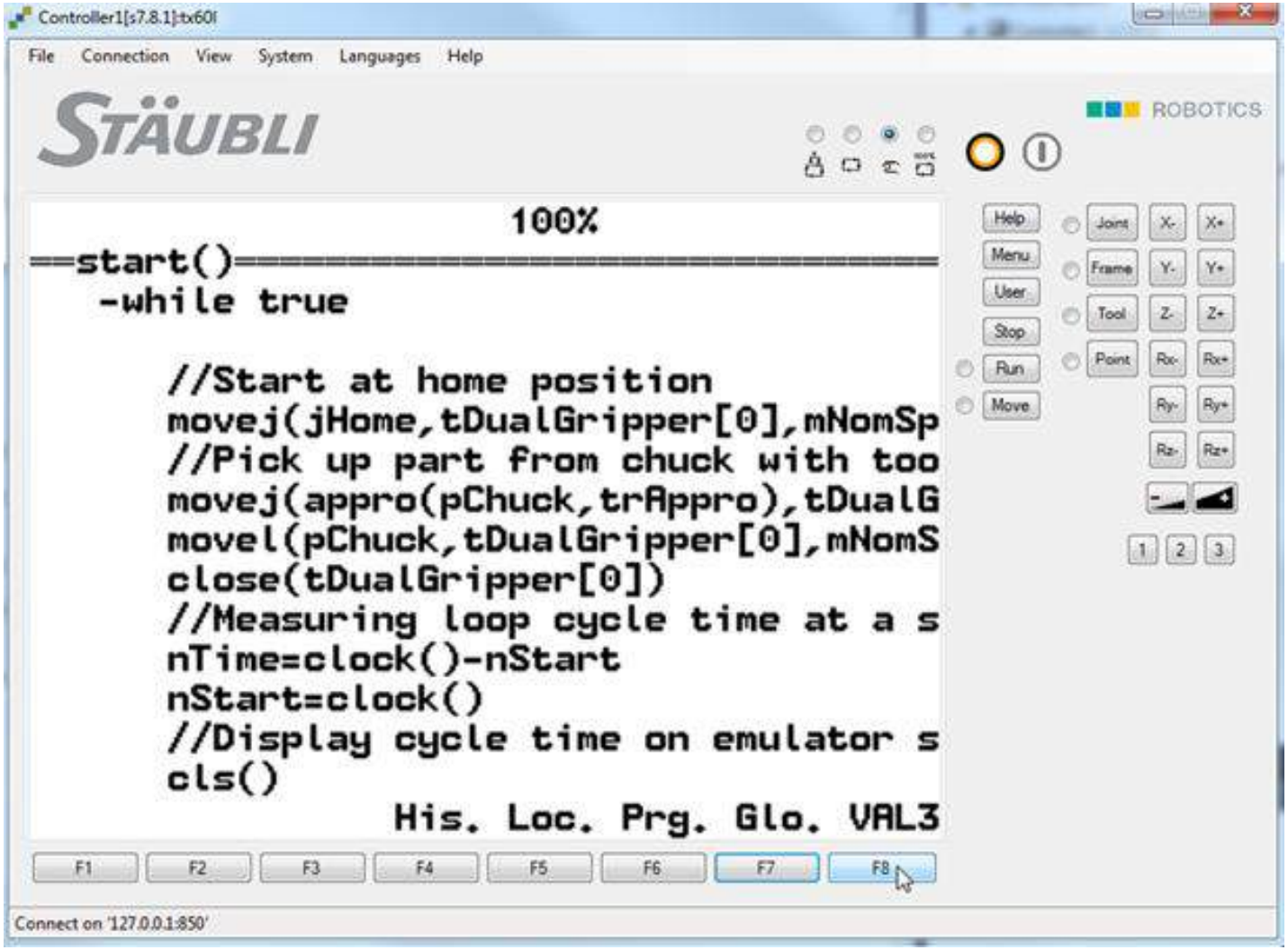
- Drzewo połączeń pozwala łatwiej zrozumieć strukturę aplikacji VAL3



EDYTOR KODU VAL 3 Z EMULATOREM (CS8)

Funkcja darmowa

- Programy VAL 3 mogą być również edytowane w emulatorze (ten sam edytor co na panelu pilota)

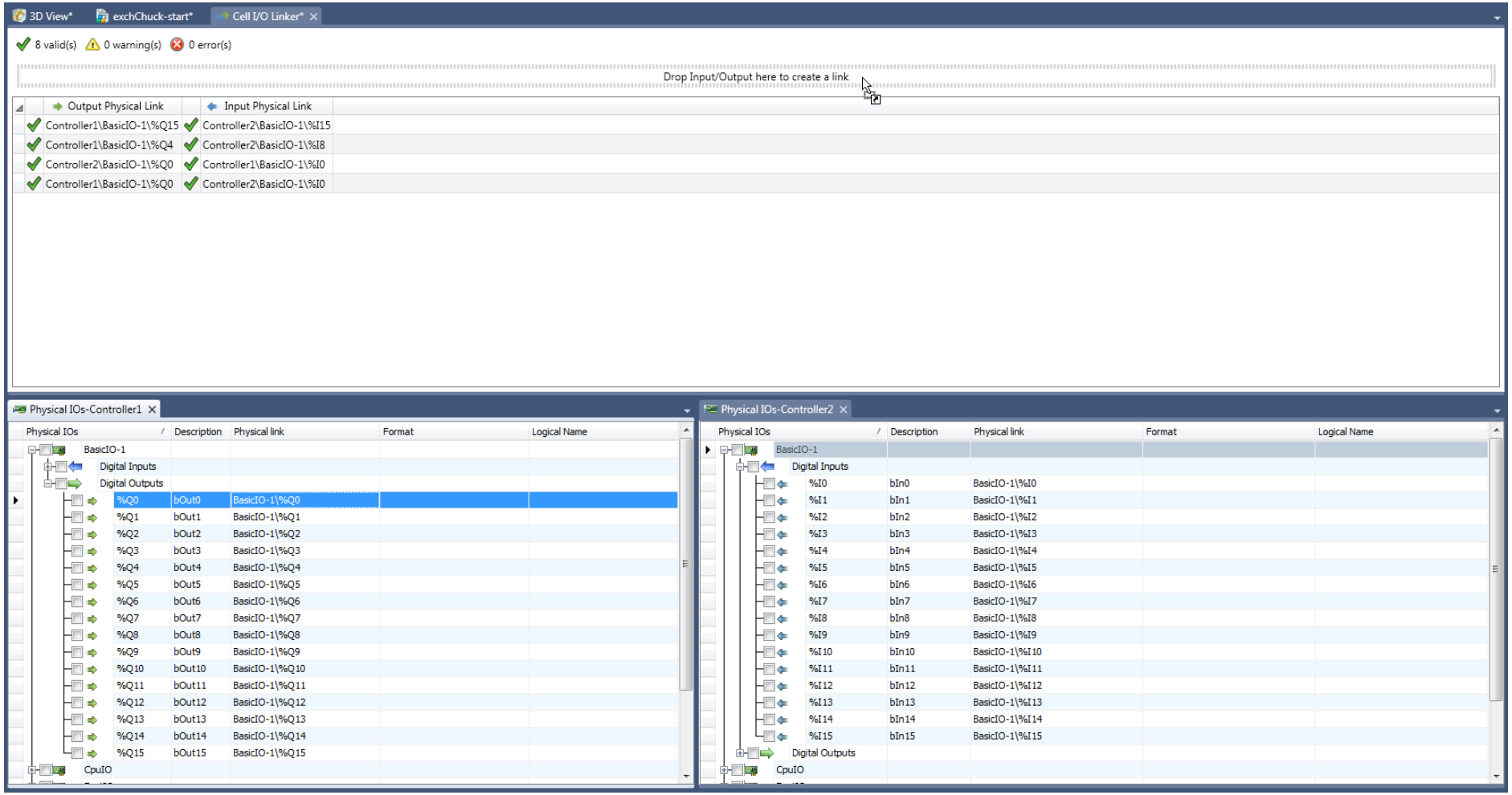


WIRTUALNE OKABLOWANIE

Development Studio

Wymagana licencja do używania tych funkcji

- Połączenia wejścia – wyjścia pomiędzy kontrolerami robotów
- Najłatwiejsza konfiguracja komunikacji robotów

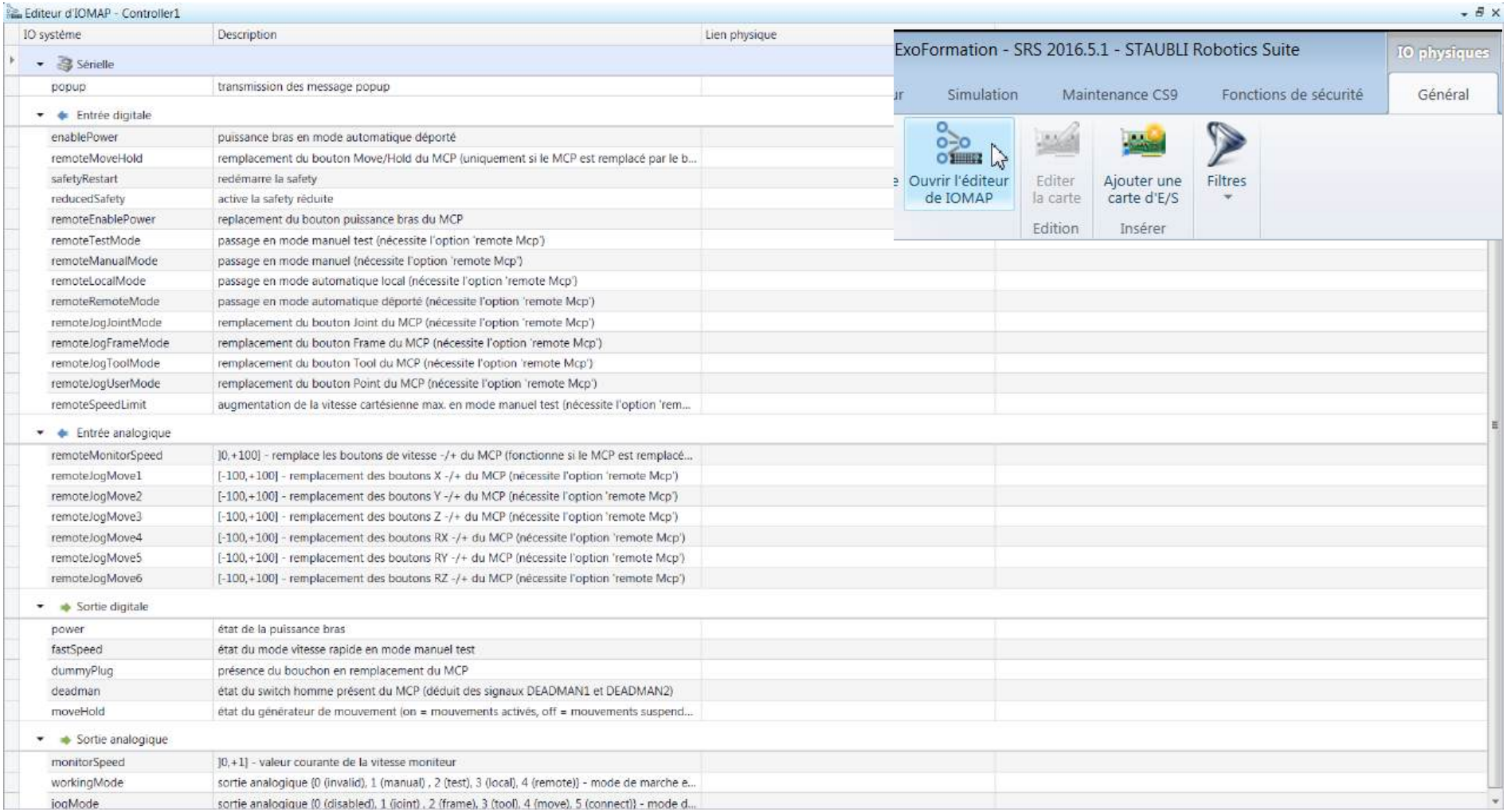


EDYTOR IOMAP

Development Studio

Wymagana licencja do używania tych funkcji

- Łączy fizyczne sygnały we/wy kontrolera z sygnałami systemowymi



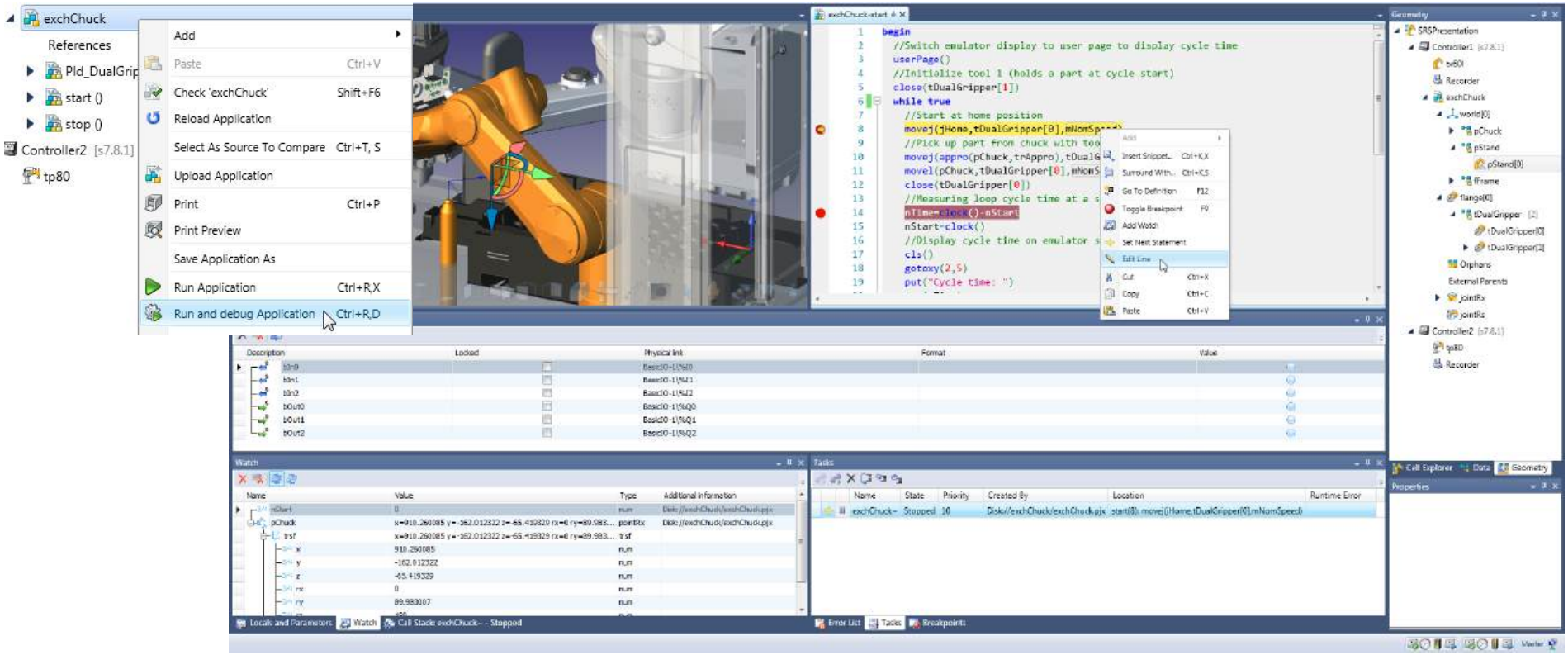
IO système	Description	Lien physique
Sérielle		
popup	transmission des message popup	
Entrée digitale		
enablePower	puissance bras en mode automatique déporté	
remoteMoveHold	remplacement du bouton Move/Hold du MCP (uniquement si le MCP est remplacé par le b...	
safetyRestart	redémarre la safety	
reducedSafety	active la safety réduite	
remoteEnablePower	replacement du bouton puissance bras du MCP	
remoteTestMode	passage en mode manuel test (nécessite l'option 'remote Mcp')	
remoteManualMode	passage en mode manuel (nécessite l'option 'remote Mcp')	
remoteLocalMode	passage en mode automatique local (nécessite l'option 'remote Mcp')	
remoteRemoteMode	passage en mode automatique déporté (nécessite l'option 'remote Mcp')	
remoteJogJointMode	remplacement du bouton Joint du MCP (nécessite l'option 'remote Mcp')	
remoteJogFrameMode	remplacement du bouton Frame du MCP (nécessite l'option 'remote Mcp')	
remoteJogToolMode	remplacement du bouton Tool du MCP (nécessite l'option 'remote Mcp')	
remoteJogUserMode	remplacement du bouton Point du MCP (nécessite l'option 'remote Mcp')	
remoteSpeedLimit	augmentation de la vitesse cartésienne max. en mode manuel test (nécessite l'option 'rem...	
Entrée analogique		
remoteMonitorSpeed	]0,+100] - remplace les boutons de vitesse -/+ du MCP (fonctionne si le MCP est remplacé...	
remoteJogMove1	[-100,+100] - remplacement des boutons X -/+ du MCP (nécessite l'option 'remote Mcp')	
remoteJogMove2	[-100,+100] - remplacement des boutons Y -/+ du MCP (nécessite l'option 'remote Mcp')	
remoteJogMove3	[-100,+100] - remplacement des boutons Z -/+ du MCP (nécessite l'option 'remote Mcp')	
remoteJogMove4	[-100,+100] - remplacement des boutons RX -/+ du MCP (nécessite l'option 'remote Mcp')	
remoteJogMove5	[-100,+100] - remplacement des boutons RY -/+ du MCP (nécessite l'option 'remote Mcp')	
remoteJogMove6	[-100,+100] - remplacement des boutons RZ -/+ du MCP (nécessite l'option 'remote Mcp')	
Sortie digitale		
power	état de la puissance bras	
fastSpeed	état du mode vitesse rapide en mode manuel test	
dummyPlug	présence du bouchon en remplacement du MCP	
deadman	état du switch homme présent du MCP (déduit des signaux DEADMAN1 et DEADMAN2)	
moveHold	état du générateur de mouvement (on = mouvements activés, off = mouvements suspend...	
Sortie analogique		
monitorSpeed	]0,+1] - valeur courante de la vitesse moniteur	
workingMode	sortie analogique {0 (invalid), 1 (manual) , 2 (test), 3 (local), 4 (remote)} - mode de marche e...	
jogMode	sortie analogique {0 (disabled), 1 (joint) , 2 (frame), 3 (tool), 4 (move), 5 (connect)} - mode d...	

ONLINE DEBUGGER

Development Studio

Wymagana licencja do używania tych funkcji

- Wykonywanie krok po kroku
- „Szpieg” na zmiennych, panel sterowania wejściami i wyjściami
- Modyfikacja kodu online
- Praca na emulowanych lub prawdziwych robotach

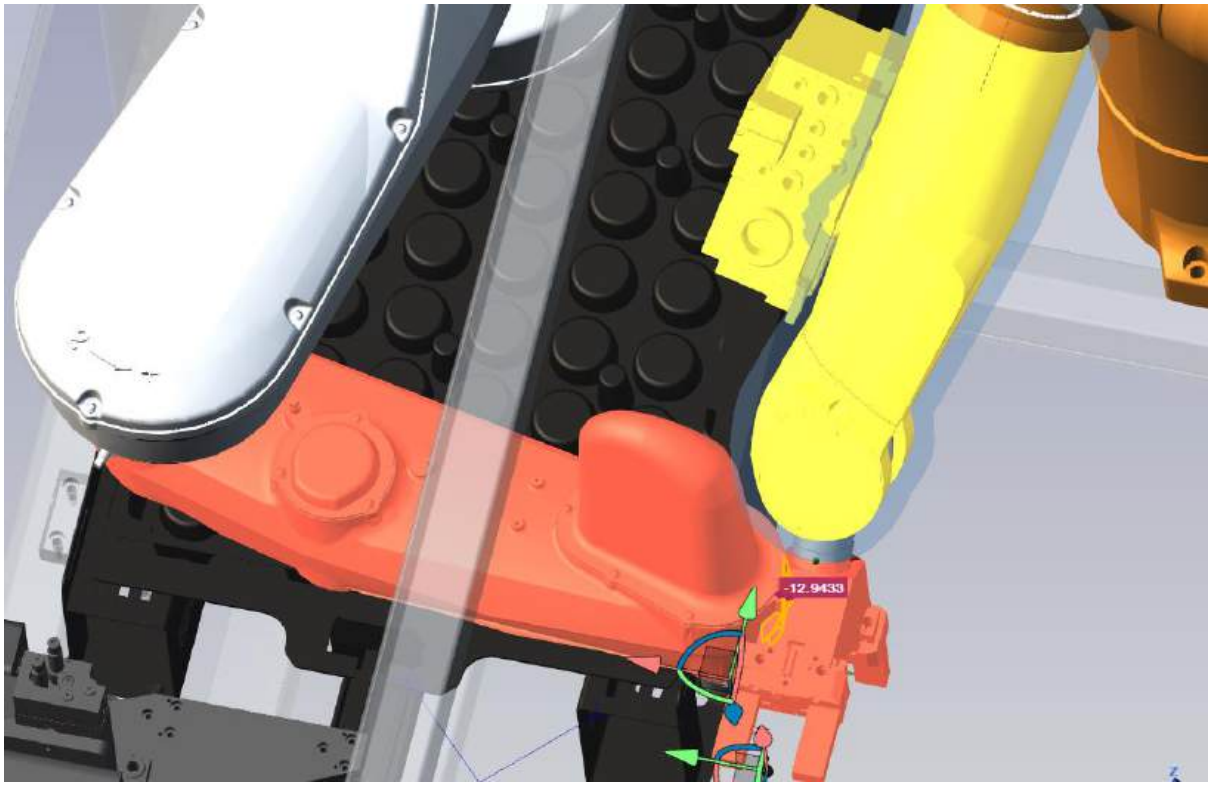


DETEKCJA KOLIZJI

Development Studio

Wymagana licencja do używania tych funkcji

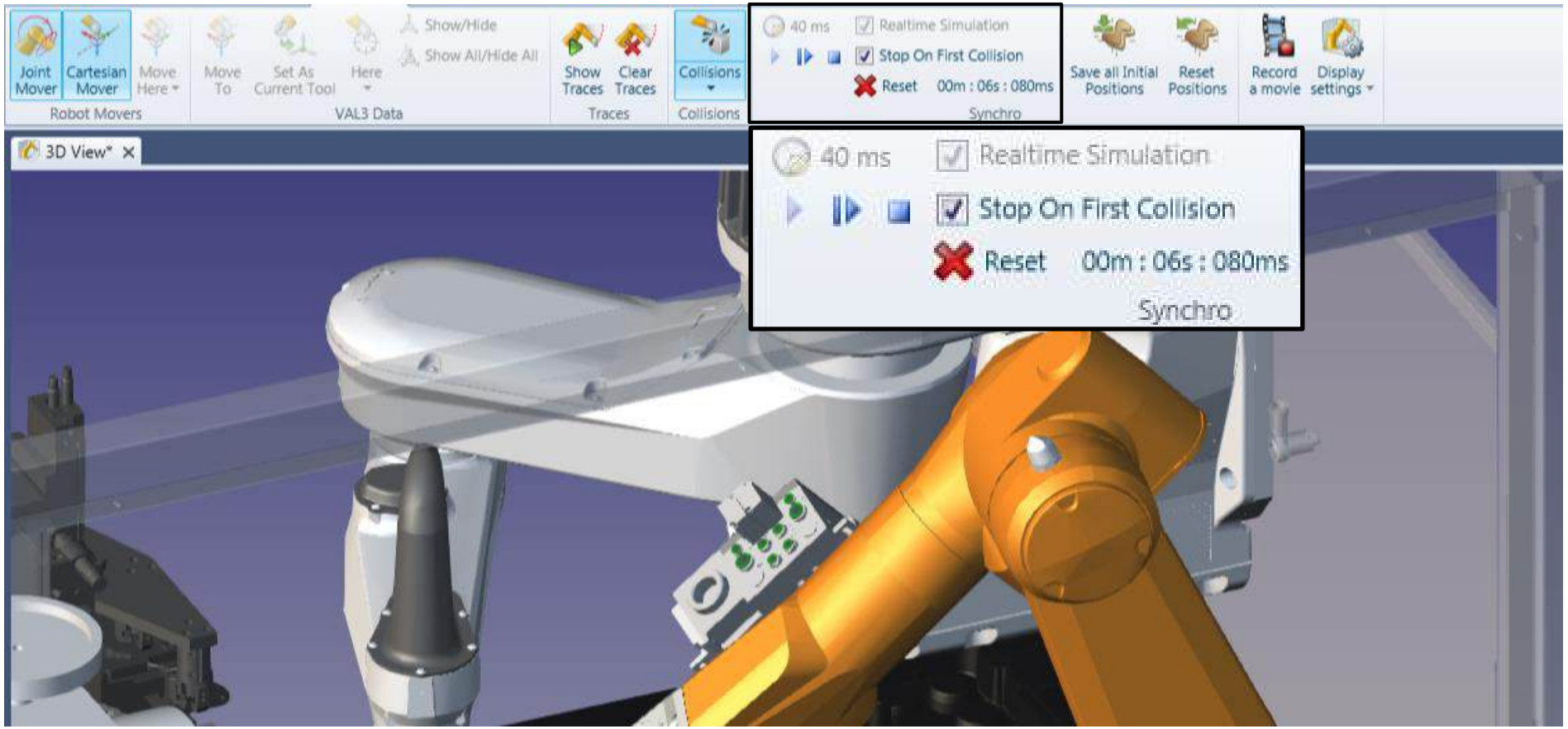
- Możliwość obejrzenia kolizji na całej scenie 3D, możliwość ustawienia wykluczeń na niektórych obiektach
- Rozmiar luzu w celu przewidywania możliwości wystąpienia kolizji
- Ruchy są wstrzymywane podczas wykonywania zaprogramowanego cyklu



REALISTYCZNY POMIAR CZASU CYKLU

Funkcja darmowa

- symulowany zegar czasu rzeczywistego
- Realistyczna trajektoria ruchu dla wykrywania kolizji i dokładnego pomiaru czasu cyklu

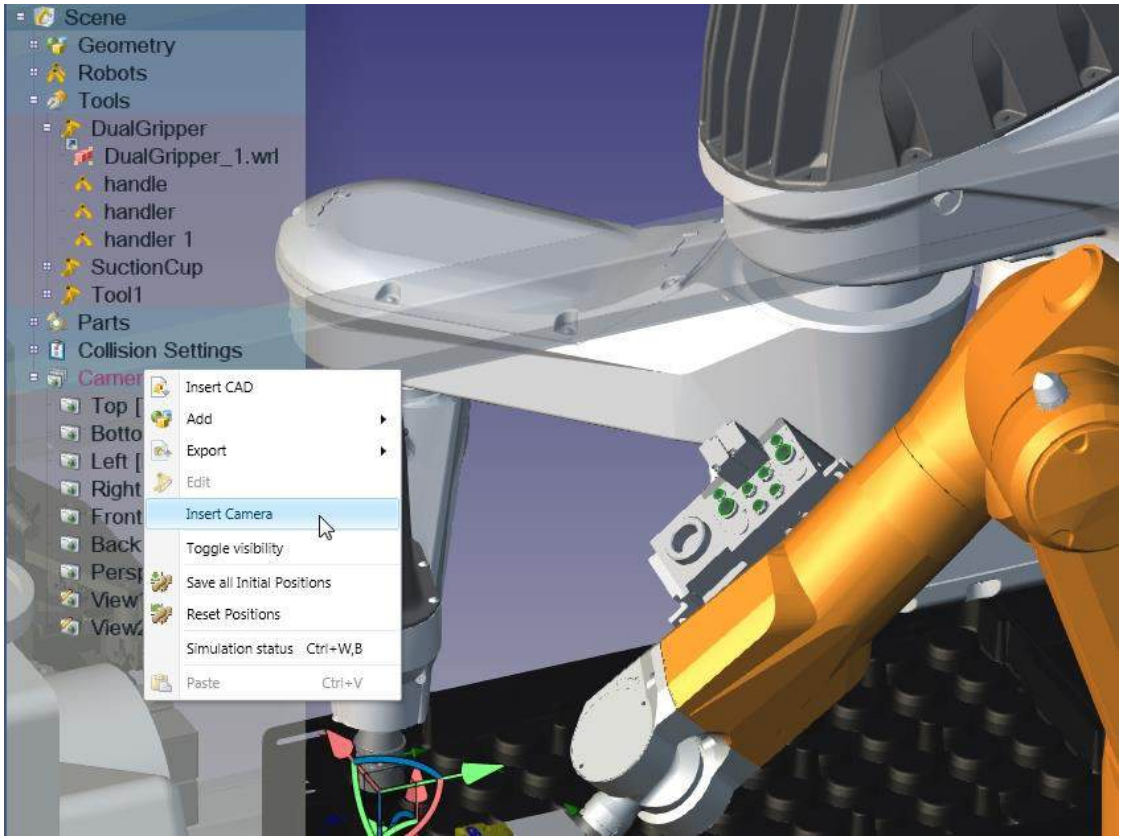


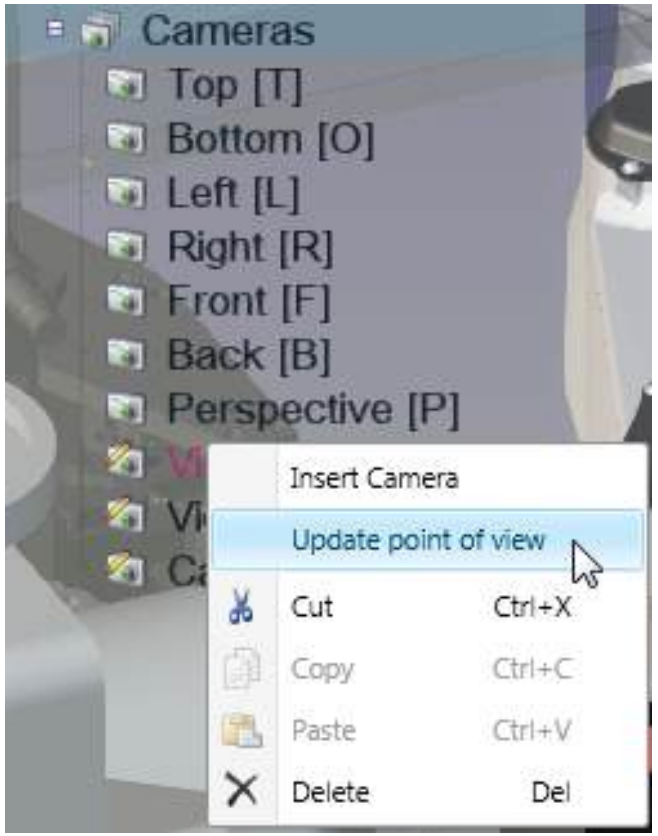
KAMERY UŻYTKOWNIKÓW

Development Studio

Wymagana licencja do używania tych funkcji

- Pozycja kamery określona przez użytkownika w scenie 3D
- Może być łatwo użyta lub zaktualizowana

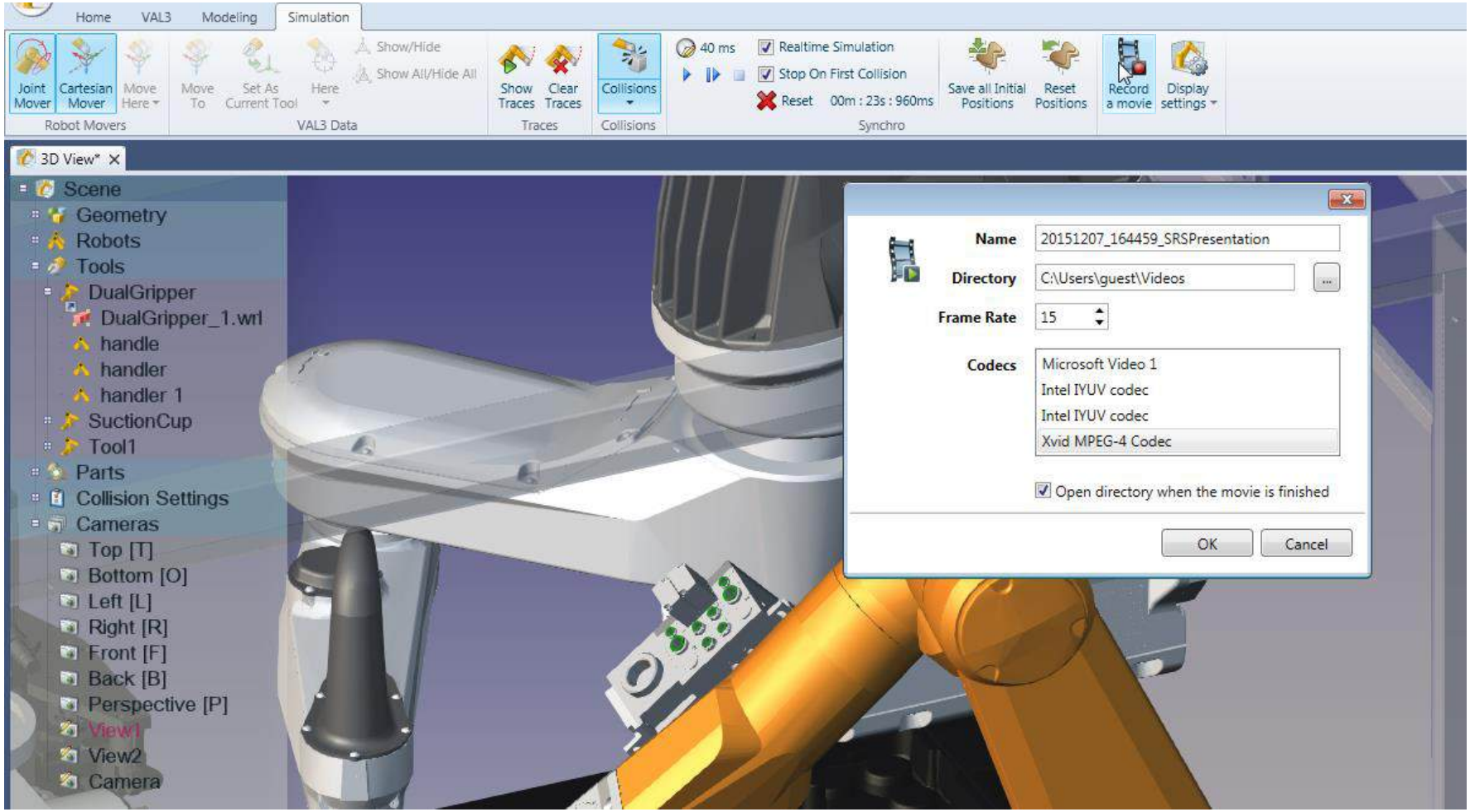




NAGRYWANIE VIDEO

Funkcja darmowa

- Scena 3D
- Standardowe generowanie plików AVI

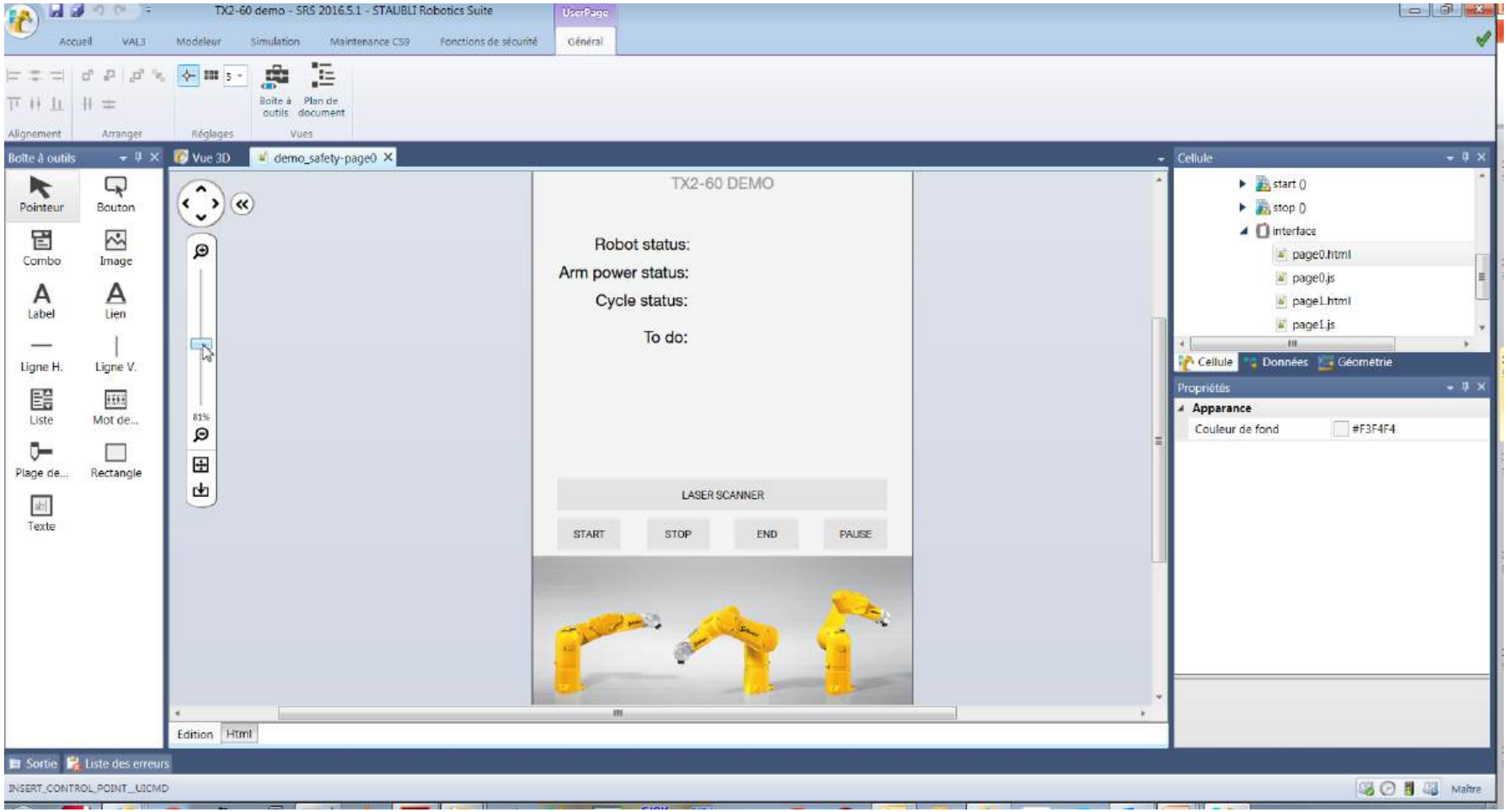


INTERFEJSY UŻYTKOWIKÓW (CS9)

Development Studio

Wymagana licencja do używania tych funkcji

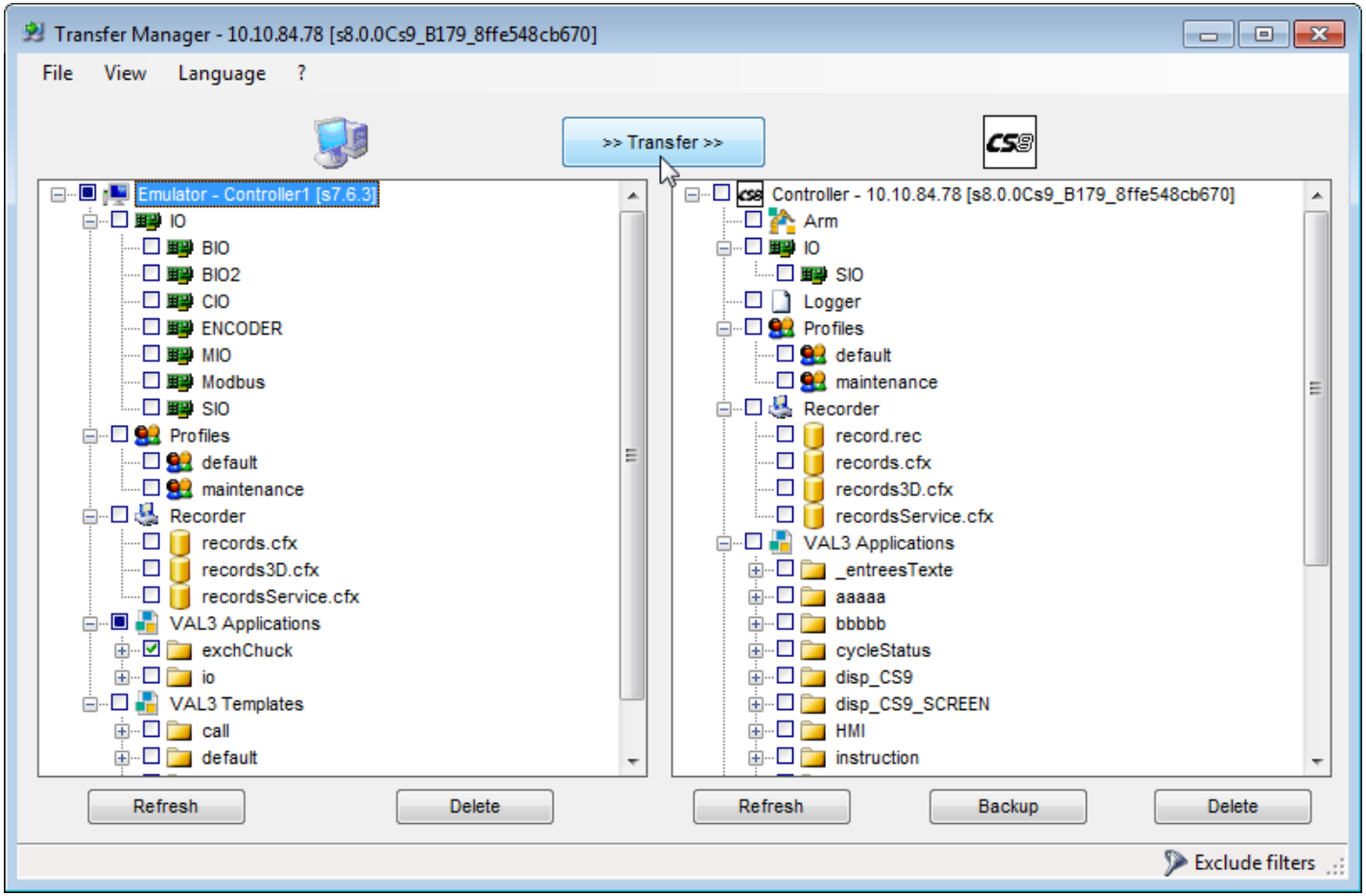
- Tworzenie graficzne stron użytkowników za pomocą dedykowanego interfejsu (proste przeciąganie i upuszczanie)
- Możliwość łączenia elementów interfejsu ze zmiennymi aplikacyjnymi VAL 3



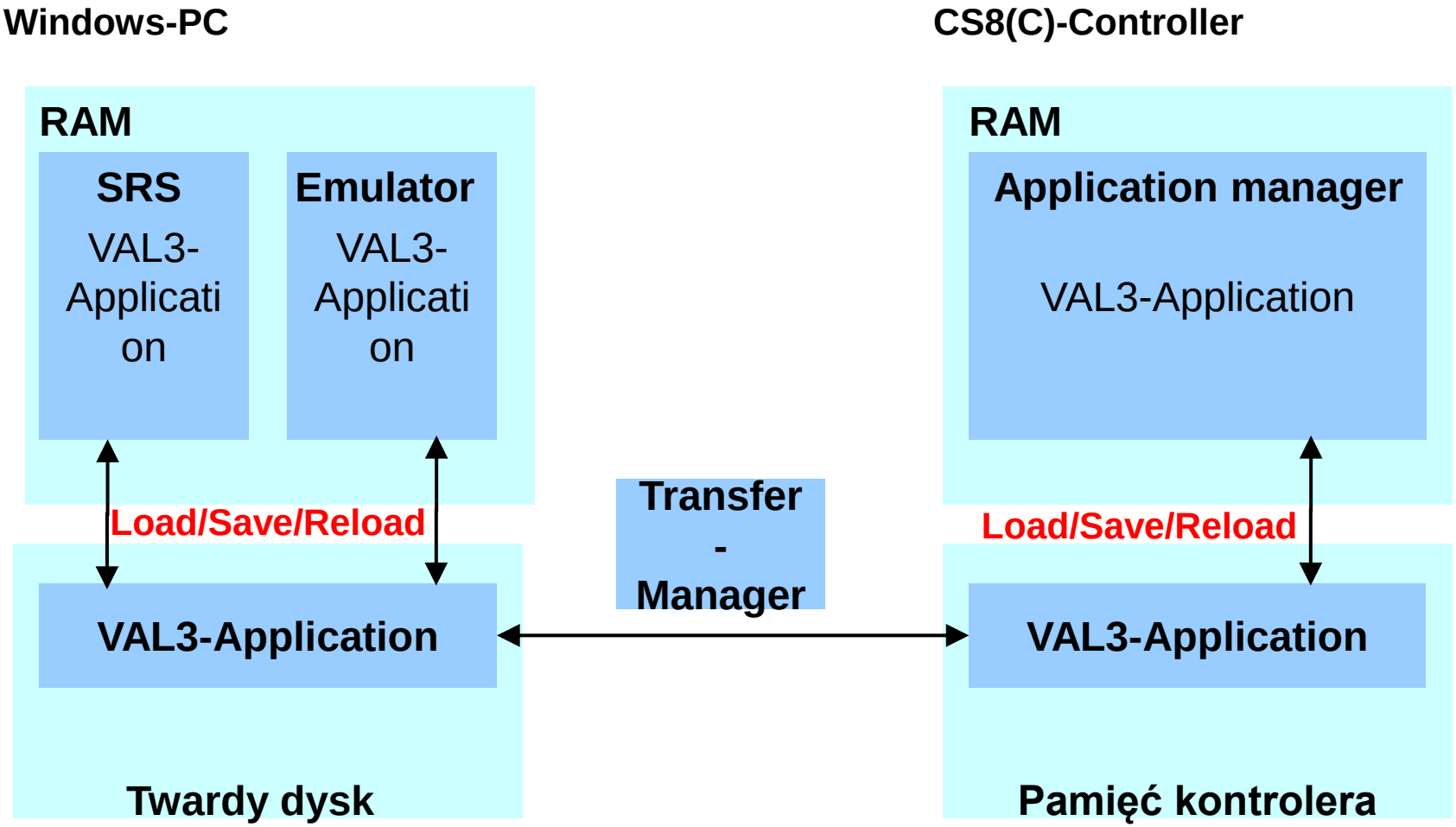
TRANSFER PLIKÓW – TRANSFER MANAGER

Funkcja darmowa

- Dwukierunkowy transfer między emulowanymi i prawdziwymi robotami
- Kopia zapasowa systemu za pomocą jednego kliknięcia



STRUKTURA PLIKÓW

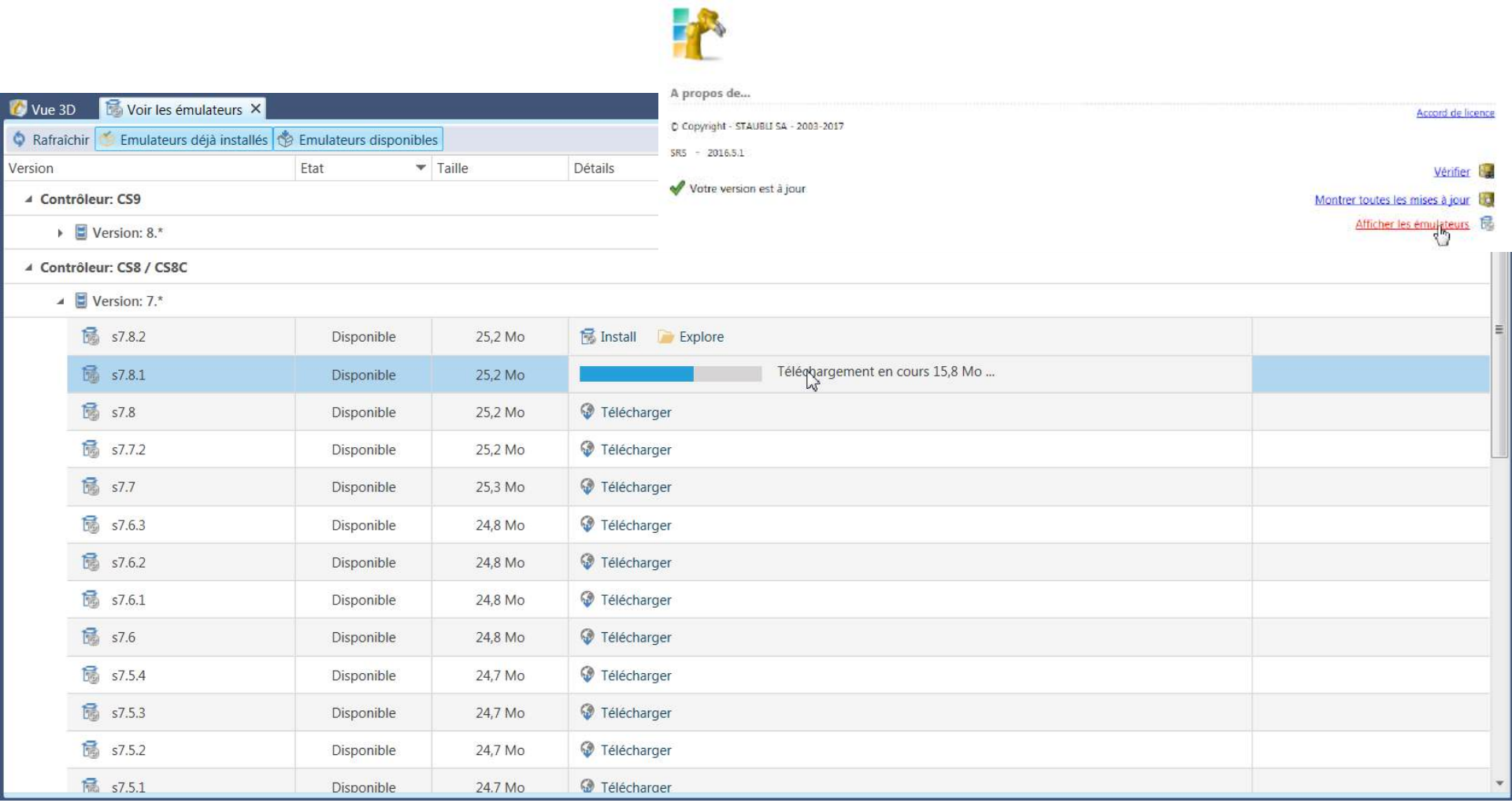


ZARZĄDZANIE EMULATOREM

Development Studio

Wymagana licencja do używania tych funkcji

- Proste zarządzanie i instalacja różnych wersji emulatorów CS8 i CS9



ZDALNY DOSTĘP – REMOTE ACCESS

Maintenance Studio

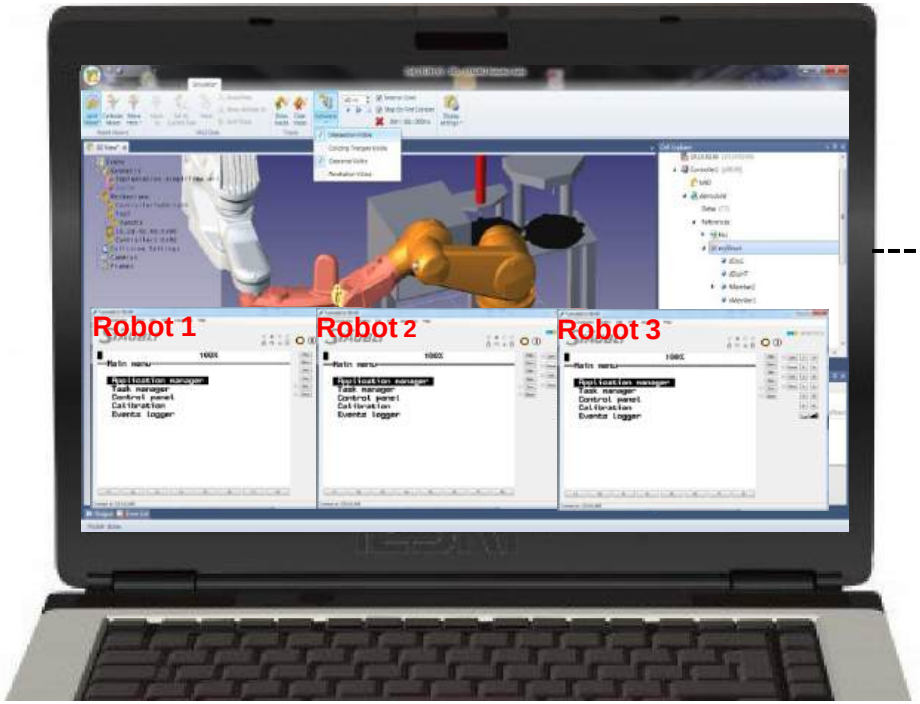
Wymagana licencja do używania tych funkcji

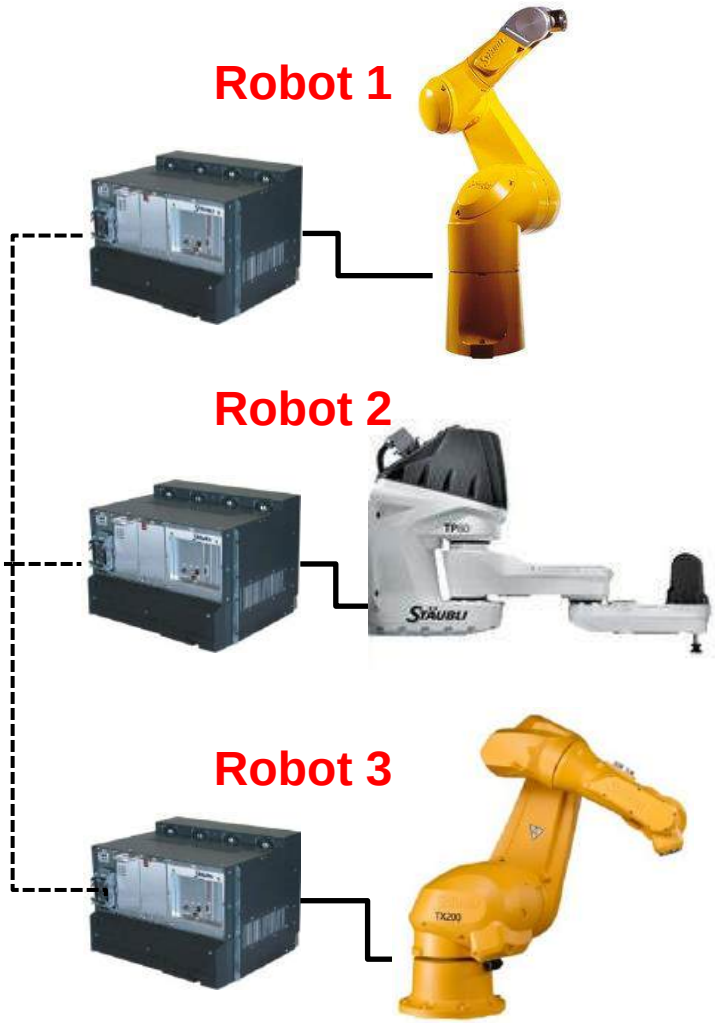
- Połączenie robotów 3D z prawdziwymi robotami

(ramiona 3D śledzą ruchy prawdziwych robotów)

- Zdalny dostęp do pilota

(logi, control panel, I/O,…)



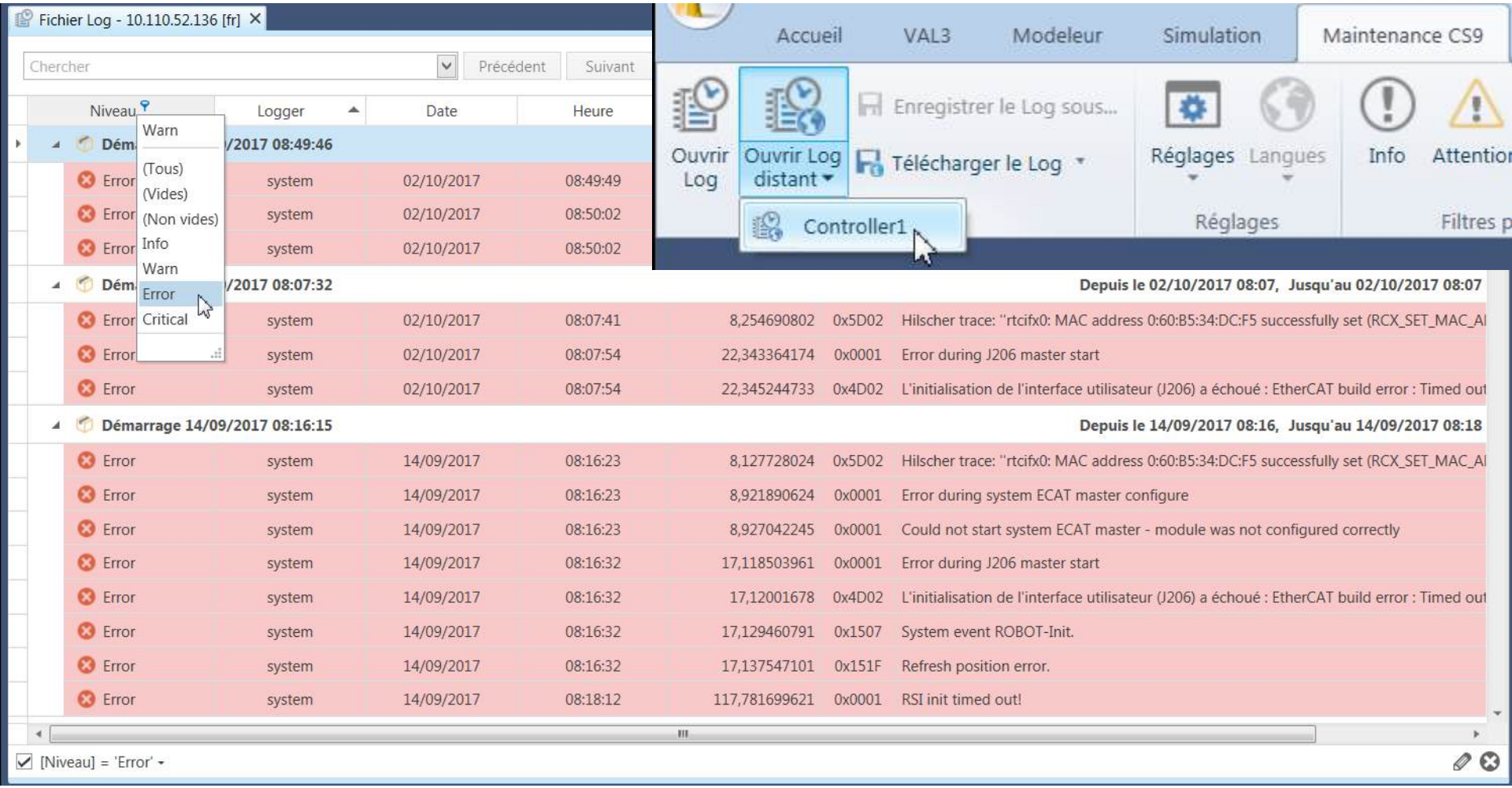


NARZĘDZIA SERWISOWE ROBOTÓW SERII CS9

Maintenance Studio

Wymagana licencja do używania tych funkcji

- Edytor logów do prostej wizualizacji komunikatów o błędach przy użyciu dostosowanych filtrów

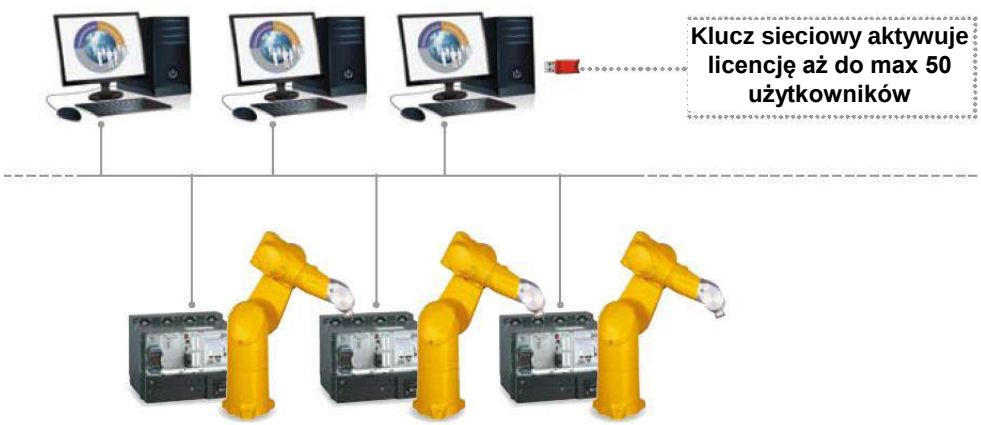


LICENCJE I AKTUALIZACJE

- Stäubli Robotics Suite zawiera domyślnie bezpłatne funkcje
- Dwie licencje, **Development Studio** i **Maintenance Studio**, aktywuje licencję poprzez klucz USB (*dongle*)
- Drobne aktualizacje są bezpłatne

Klucz lokalny, 1 licencja-1 klucz

Klucz sieciowy, kilka licencji na 1 kluczu



BLENDING – ćwiczenie

Ćwiczenie nr 5: Blending

Proszę zmodyfikować program z ćwiczenia nr 3 (zadanie 1), zmieniając ustawienia Blending-u, zgodnie z poleceniami w zadaniu 2, wykorzystując do tego celu program SRS.

Proszę dodać odpowiednie zmienne.

TX-TS-RX-RS-TP/**CS8C**

TX2/**CS9**

CZĘŚĆ 2 – USER CS9

OPTIMIZE LAB - RECORDER

RECORDER – NAGRYWANIE TRAJEKTORII

Konfiguracja nagrywania

Step 1

- Plik konfiguracyjny generowany za pomocą **Optimize Lab**(1)
- Plik wgrywany na kontroler przez :Optimize Lab(1), SRS(1), FTP, lub na pamięć USB podłączaną do kontrolera

Nagrywanie trajektorii

Step 2

- Uruchomienie robota w normalnym cyklu produkcyjnym
- Nagrywanie trajektorii jest uruchamianie i/lub zatrzymywane z teachpendanta

Zapis i kopiowanie nagrań

Step 3

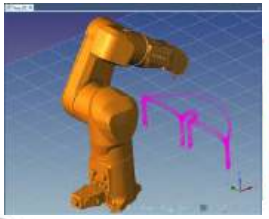
- Wyniki są zapisywane na pamięci USB lub wewnętrznej pamięci kontrolera, skąd mogą być skopiowane przez Optimize Lab(1), SRS(1), lub FTP

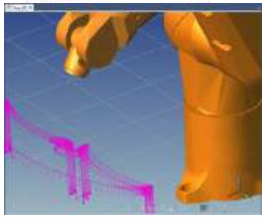
Analiza nagrań

Step 4
1

- Analiza nagrań odbywa się za pomocą programu Optimize Lab(1). Pozwala na określenie wpływu wykonywanej trajektorii na stan ramienia (dostosowanie serwisu, optymalizacja czasu cyklu)
- Optimize Lab(1) umożliwia utworzenie wykresów z wartościami np. prędkości, przyspieszeń, prądów, sił i momentów na poszczególnych osiach

Wizualizacja wyników





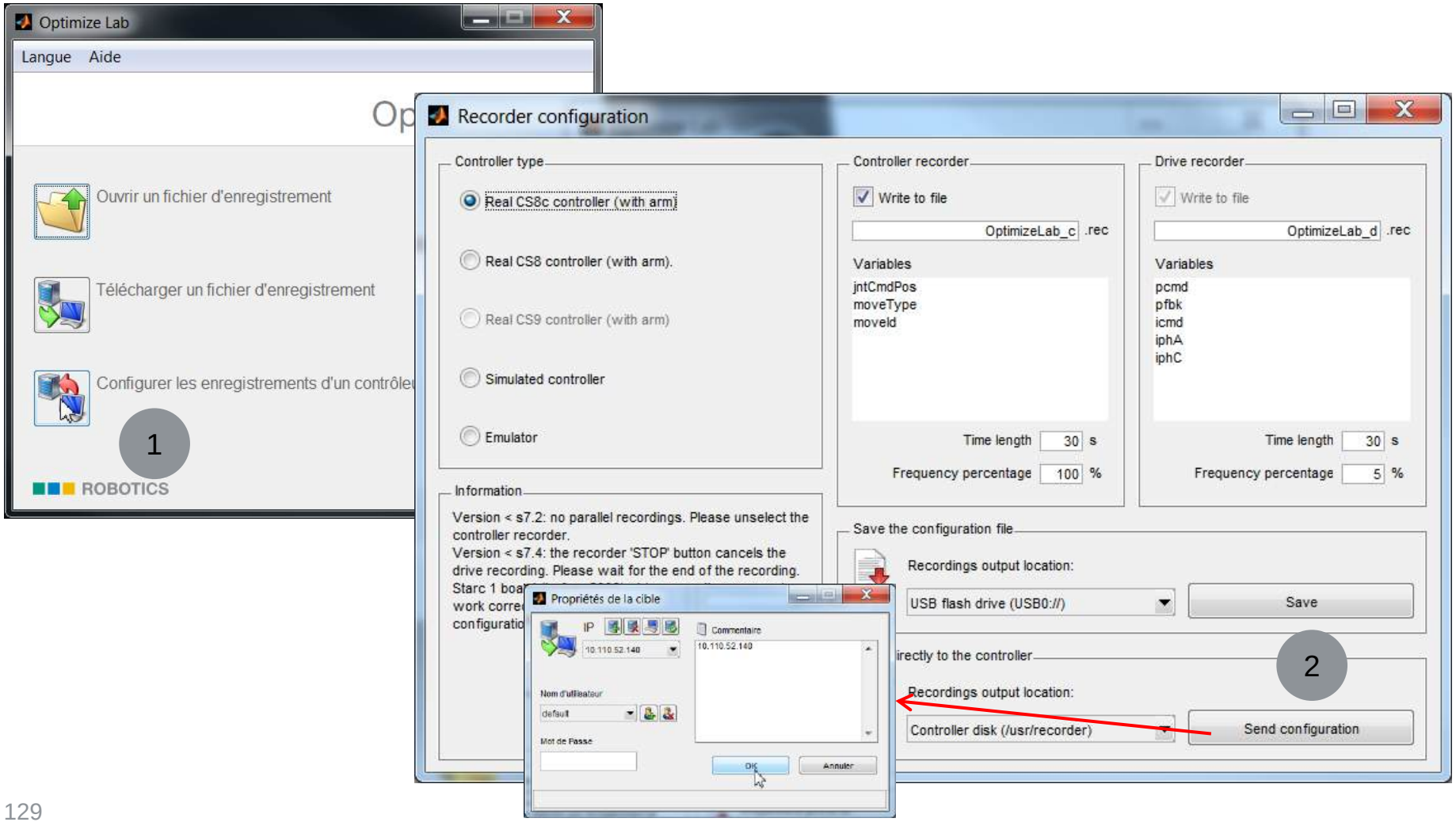
Step 4
2

- Trajektoria może być wyświetlona w 3D w SRS(1)

OPTIMIZE LAB CONFIGURATION

Optimize Lab

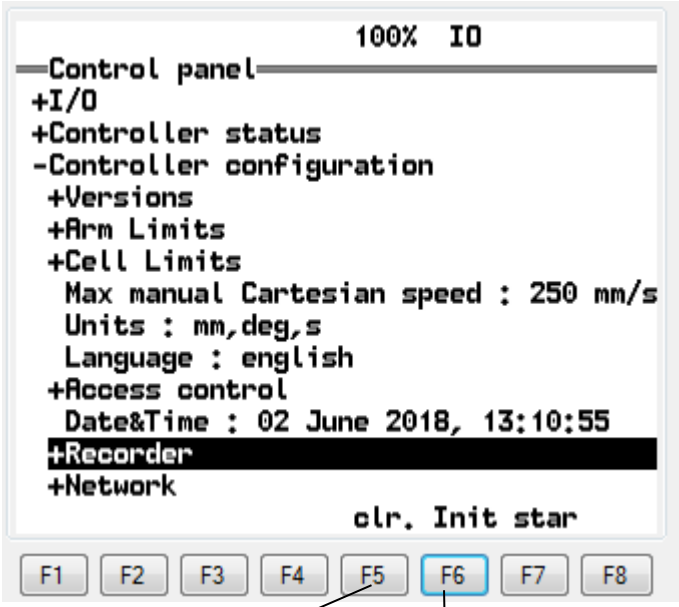
1. Parametry konfiguracji recordera
2. Wyślij plik konfiguracyjny na kontroler lub zapisz go na pamięci USB



KONFIGURACJA ŁADOWANA PRZEZ USB (CS8C)

- W przypadku użycia kontrolera CS8C, pliki wynikowe będą zapisane na pamięci USB

1 Podłącz pamięć USB do lewego portu (na root pamięci USB musi znajdować się plik konfiguracyjny: **records.cfx**)

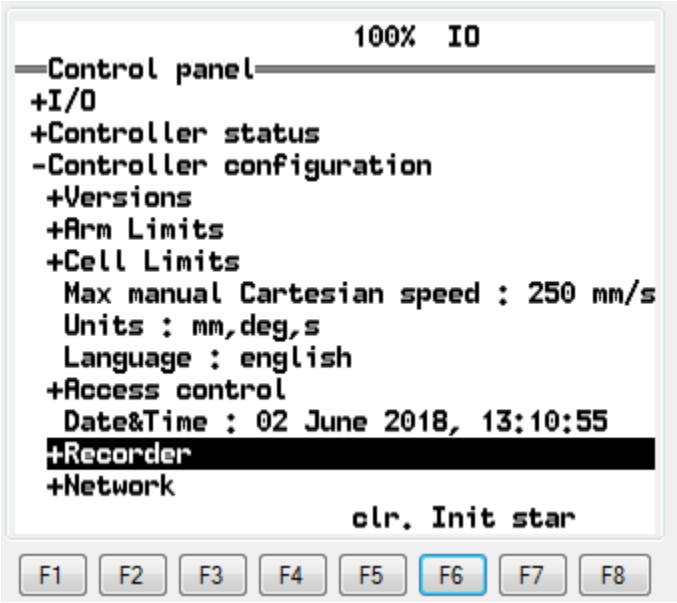


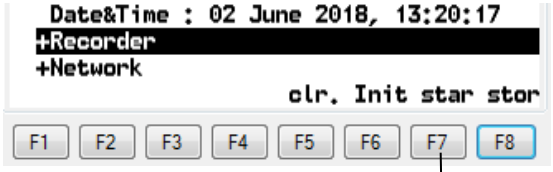
2

clr: Kasuje poprzednie nagrane pliki trajektorii

Init: inicjalizacja pliku konfiguracyjnego

RECORD (CS8C)



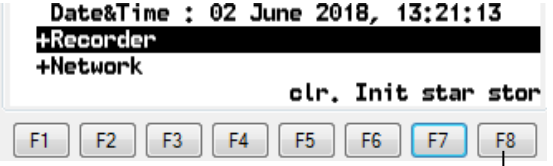


Naciśnij « **start** » aby rozpocząć nagrywanie tajektorii

1

« **stop** » ręczne zatrzymanie nagrywania

2



« **store** » export / zapis nagranej trajektorii na pamięć USB (*records.rec*)

3

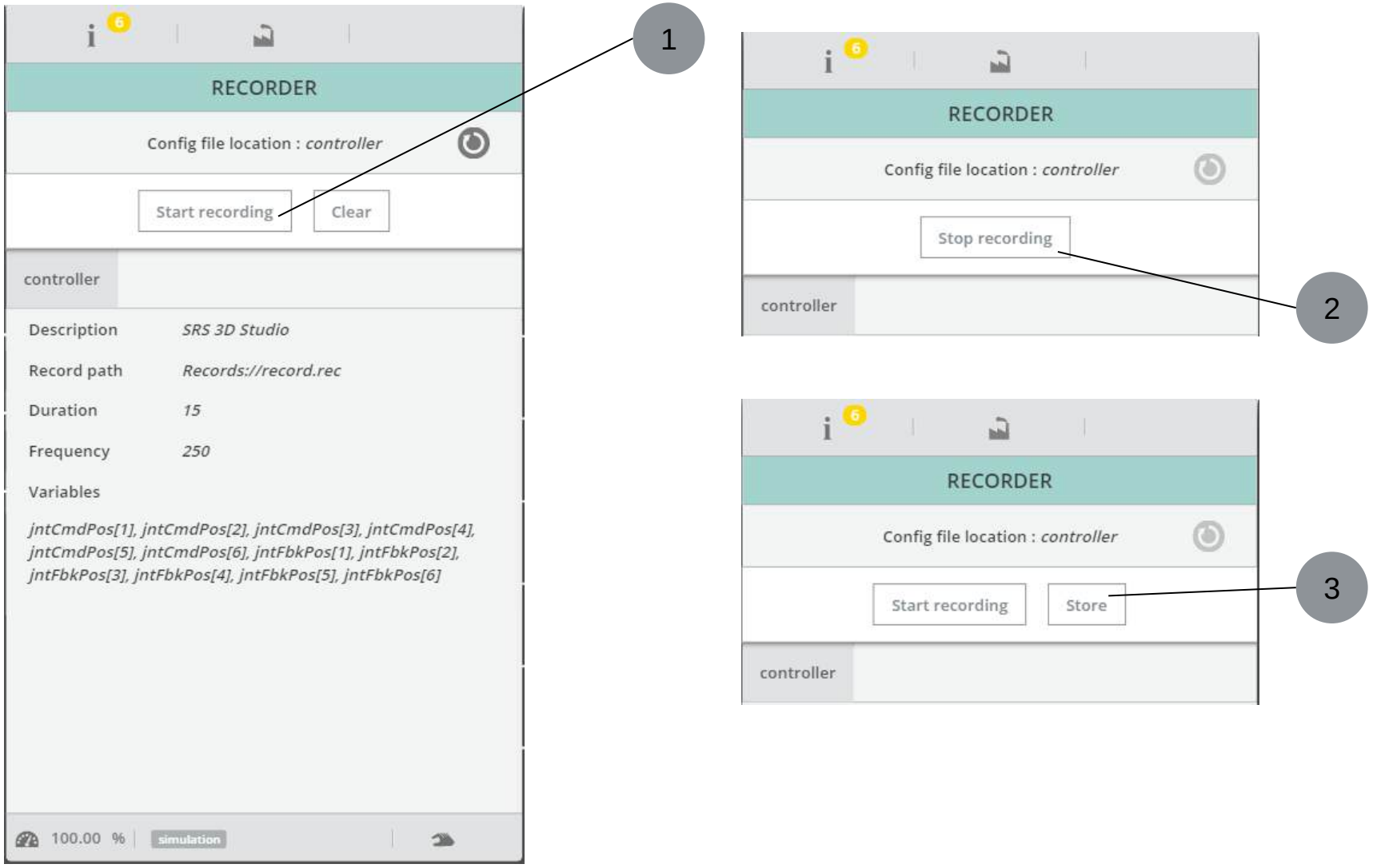
RECORD (CS9)

- W przypadku użycia kontrolera CS9, pliki wynikowe będą domyślnie zapisane na pamięci USB

1 Podłącz pamięć USB do portu (na root pamięci USB musi znajdować się plik konfiguracyjny: **records.cfx**)

Dane konfiguracji

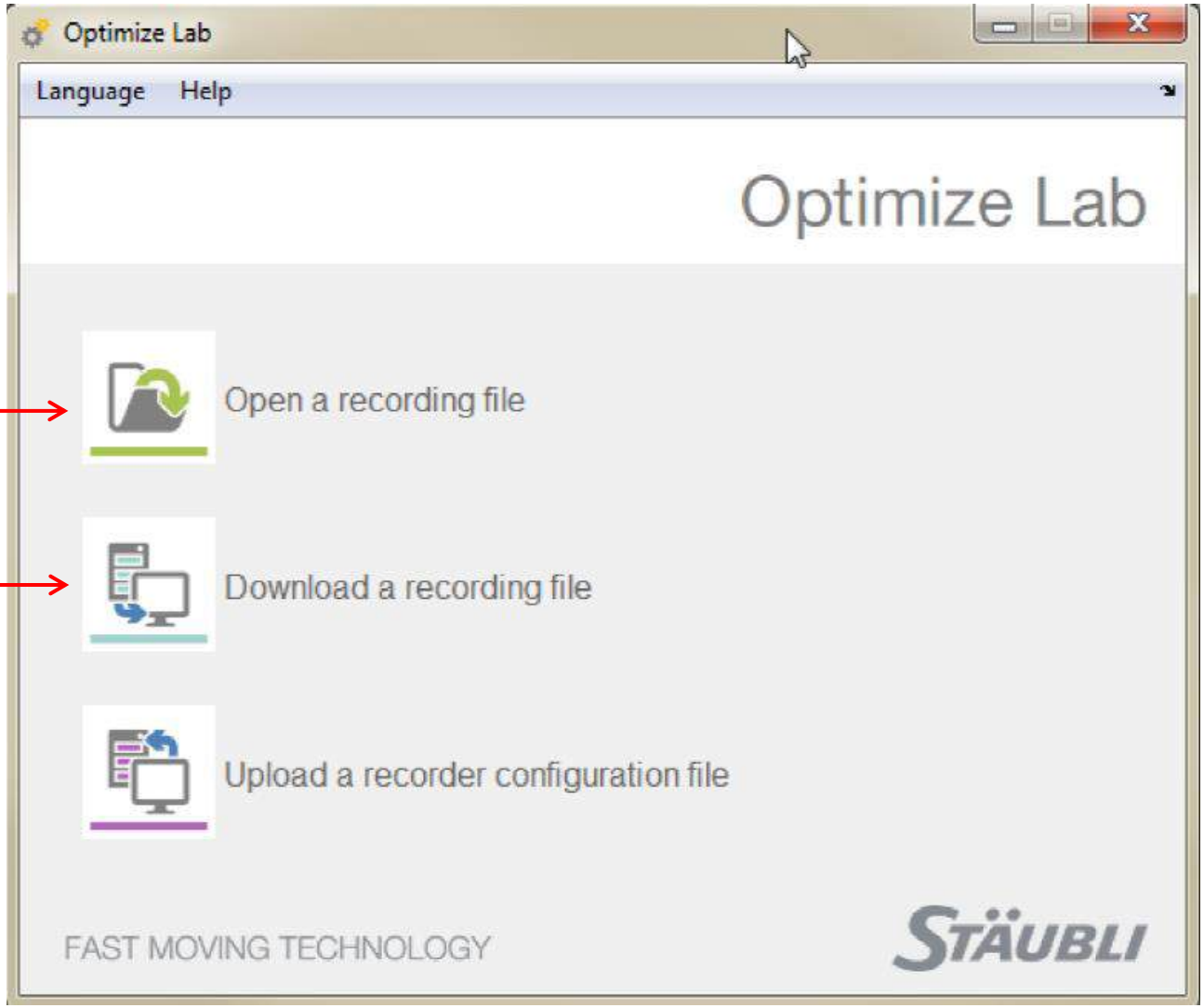
RECORD (CS9) – C.D.



ANALIZA TRAJEKTORII PRACY ROBOTA PRZY UŻYCIU PROGRAMU OPTIMIZE LAB

Jeżeli plik do analizy znajduje się na pamięci USB

Jeżeli plik do analizy znajduje się na dysku kontrolera

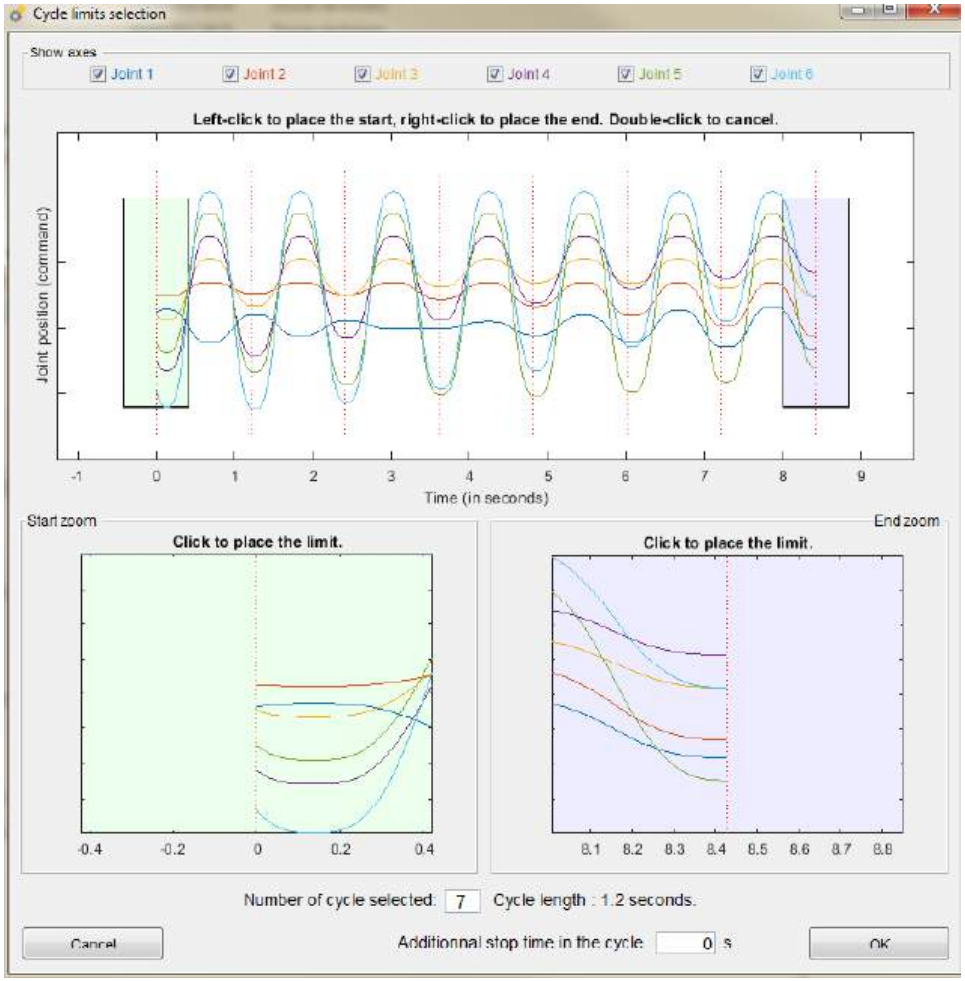


WYBÓR CYKLU

Optimize Lab

- Jeśli cykl się powtarza, wybór danych odbywa się automatycznie na krzywych położenia kątowego
- Możliwa jest modyfikacja wyboru

Kliknij na lewo aby zmienić pozycję początkową

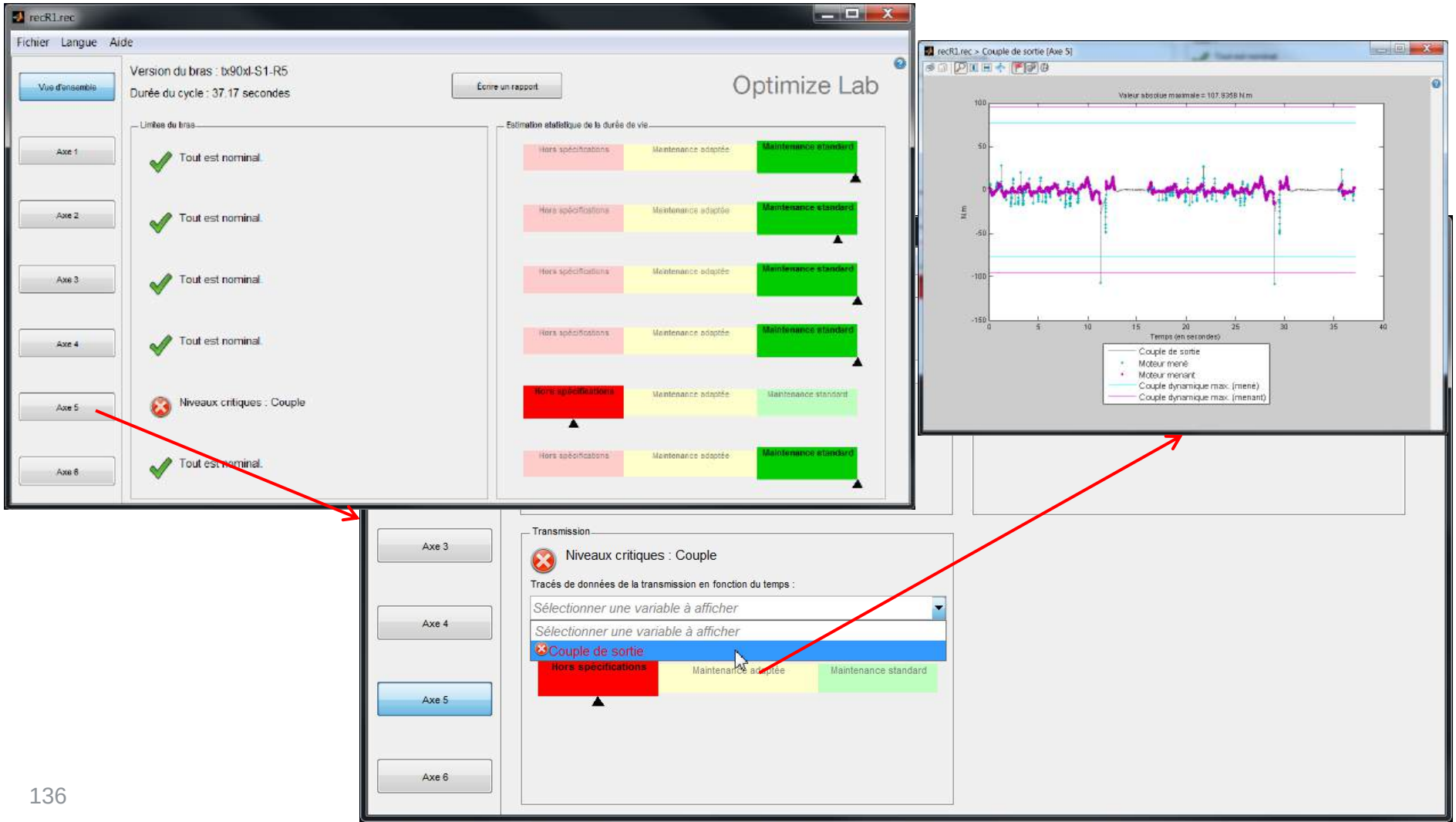


Kliknij na prawo aby zmienić pozycję końcową

ANALIZA WYNIKU

Optimize Lab

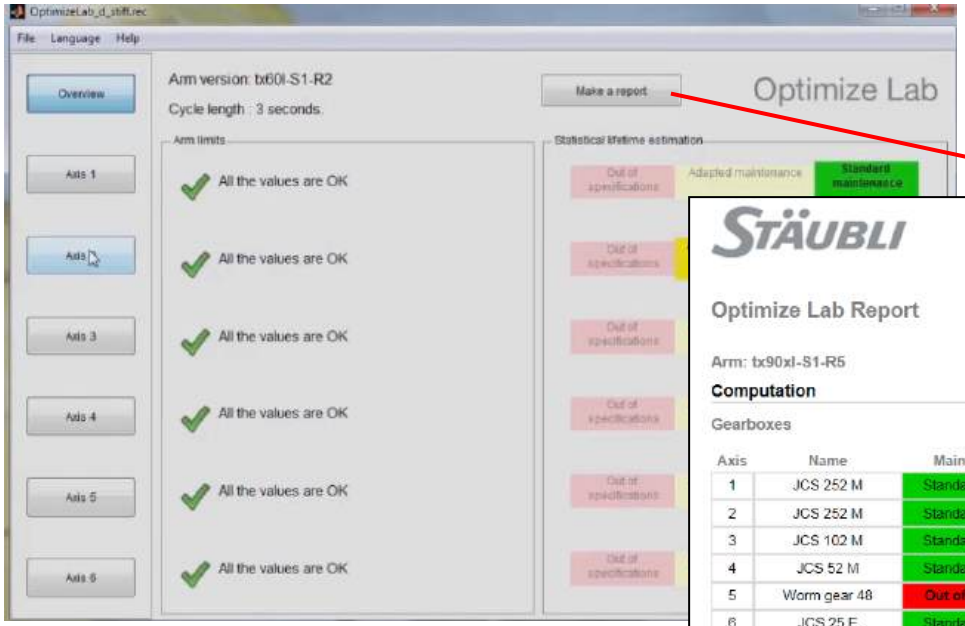
- **Optimize Lab** po analizie daje wynik dotyczący obciążenia każdej osi robota i pokazuje użytkownikowi parametr i wykres najlepiej odzwierciedlające problem



RAPORT Z ANALIZY

Optimize Lab

- Optimize Lab może wygenerować raport z przeprowadzonej analizy (format XML)



STÄUBLI ROBOTICS

Optimize Lab Report

Arm: tx90xl-S1-R5

Computation

Gearboxes

Axis	Name	Maintenance level
1	JCS 252 M	Standard maintenance
2	JCS 252 M	Standard maintenance
3	JCS 102 M	Standard maintenance
4	JCS 52 M	Standard maintenance
5	Worm gear 48	Out of specifications
6	JCS 25 E	Standard maintenance

Limits

103 limits were checked. No warning,1 critical parameter.

Axis	Device	Level	Limit name	Maximum value
5	gearbox	Critical	Max. dynamic torque (led)	141 %

Figures

Output torque axis 5

NAGRYWANIE TRAJEKTORII – ćwiczenie

Ćwiczenie nr 6: Nagrywanie trajektorii pracy robota

Proszę zapoznać się z filmem instruktażowym **OptimizeLab**

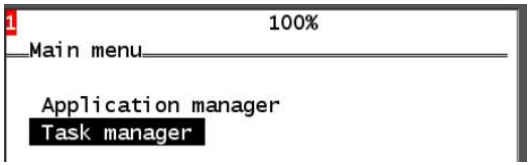
1. Proszę wgrać odpowiedni plik konfiguracyjny na pendrive'a.
2. Proszę uruchomić program z zadania1 w trybie automatycznym i nagrać trajektorię (zmieniając podparametry w parametrze prędkości – np. wł/wył blending, zwiększyć podparametr przyspieszenia)

CZĘŚĆ 2 – USER CS9

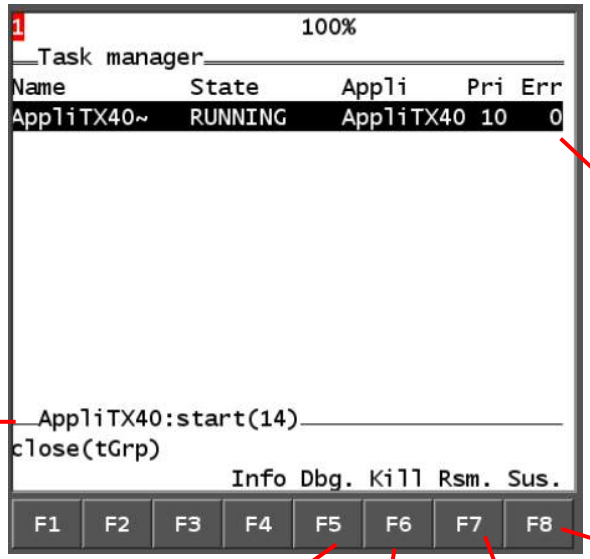
TASK MANAGER

TASK MANAGER

1 Przycisk na SP2



Podczas uruchomionej aplikacji widać jej status w « **task manager** »



Kod błędu <> 0

Priorytet zadania/taska (1-100)

Numer aktualnie wykonywanej linii programu

Aktualnie wykonywania linia programu – odświeżanie co 200 ms

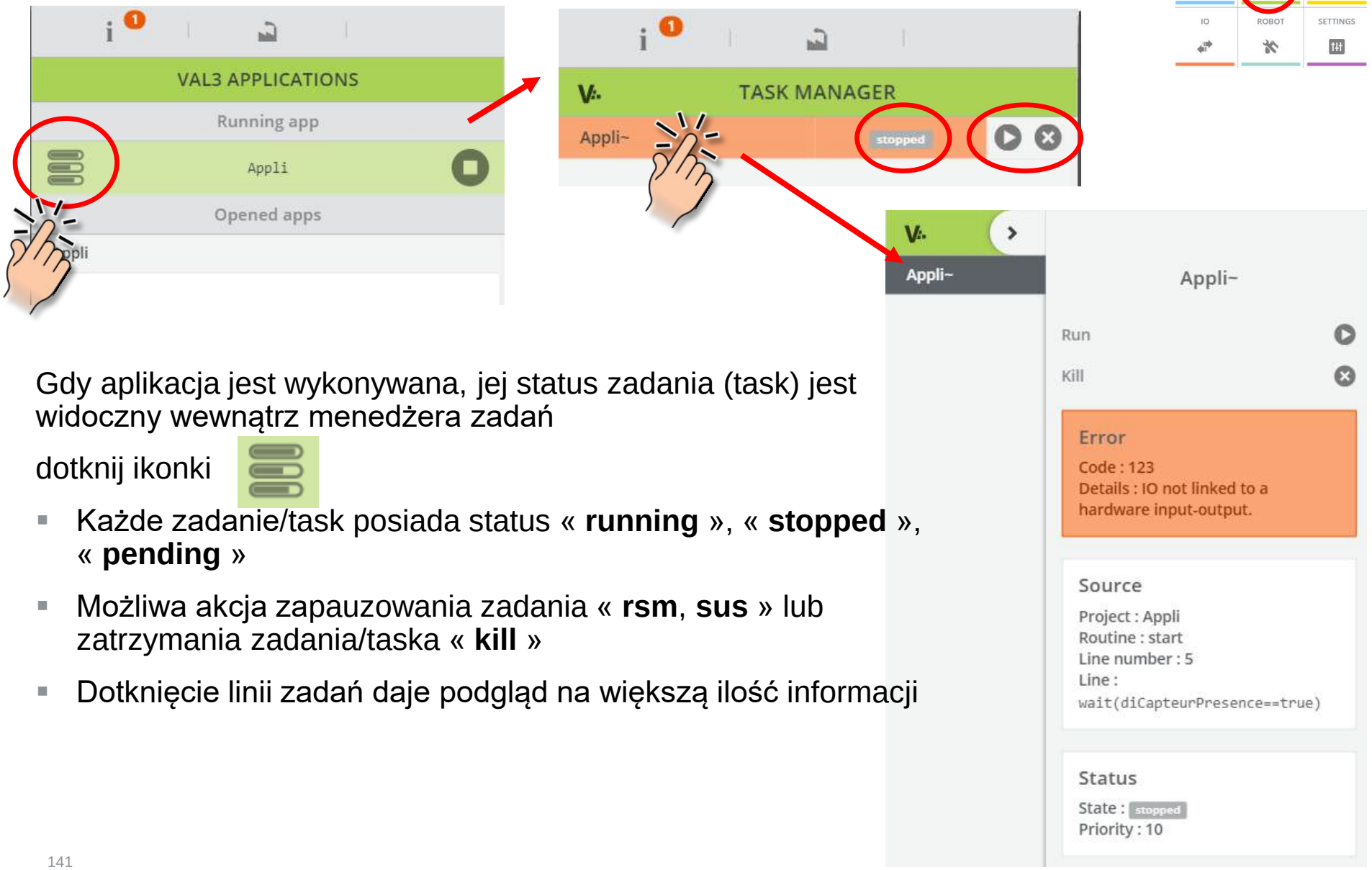
Wstrzymanie uruchomionego zadania/taska

Zatrzymaj zadanie/taska & rozpocznij debugging

Zatrzymaj zadanie/taska

Wznowienie wstrzymanego zadania/taska

TASK MANAGER



Gdy aplikacja jest wykonywana, jej status zadania (task) jest widoczny wewnątrz menedżera zadań

dotknij ikonki

- Każde zadanie/task posiada status « **running** », « **stopped** », « **pending** »
- Możliwa akcja zapauzowania zadania « **rsm**, **sus** » lub zatrzymania zadania/taska « **kill** »
- Dotknięcie linii zadań daje podgląd na większą ilość informacji

DEBUGGER

Aby wyjść

- Pozwala uruchomić program krok po kroku i wyświetlić/modyfikować wartości zmiennych

aktualnie wykonywana linia

Breakpoint

```
                                100%
==pick(point pt,num nDistance)==========
 -begin
 >   io:valve1=true
     trApp={0,0,-nDistance,0,0,0}
*    movej(appro(pt,trApp),tlGrip,nom_spe
     movel(pt,tlGrip,nom_speed)
     close(tlGrip)
     movel(appro(pt,trApp),tlGrip,nom_spe
  end


Bpts ;    Data ->*   {}*  {*}  Rsm. Save
```

Dodawanie/Kasowanie Breakpoint'ów

Komentarze

Wyświetla zmienne

Wykonaj kolejną linię

ext step (CALL)

int step. (CALL)

Wznowienie taska

TASK MANAGER I DEBUGGING – ćwiczenie

Ćwiczenie nr 7: Task Manager i Debugging

1. Proszę wstawić do programu Brakepoint a następnie uruchomić program.
2. Proszę przejść do Task Managera i sprawdzić stan uruchomionych programów.
3. Proszę wejść do debuggingu programu i sprawdzić stan zmiennych.
4. Proszę uruchomić program krok po kroku.

CZĘŚĆ 2 – USER CS9

OBSŁUGA I/O

KOMUNIKACJA CS9 Z INNYMI URZĄDZENIAMI

Sterowanie silnikami ramienia

Sygnały z enkoderów

Fieldbus:
Profibus …

HMI / maszyny / PLC …

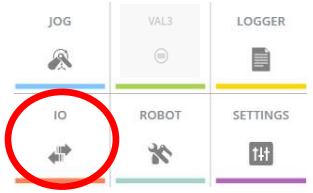
Komunikacja przez interfejs szeregowy RS232 / RS422

Komunikacja TCP/IP

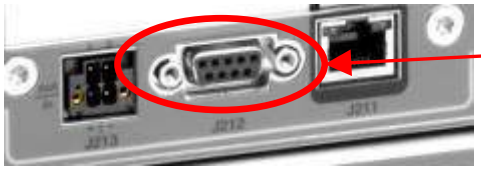
Ethernet Fieldbus (slave): Profinet, Ethercat, Modbus, …

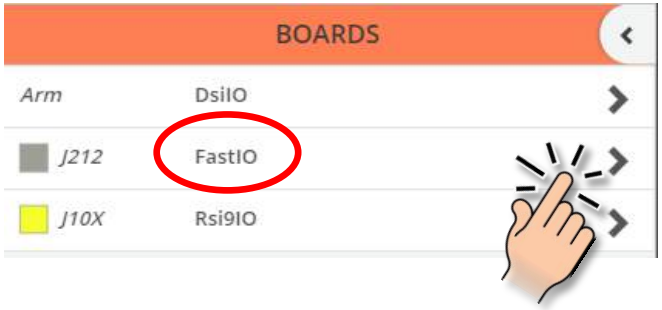
STANDARDOWE WEJŚCIA / WYJŚCIA

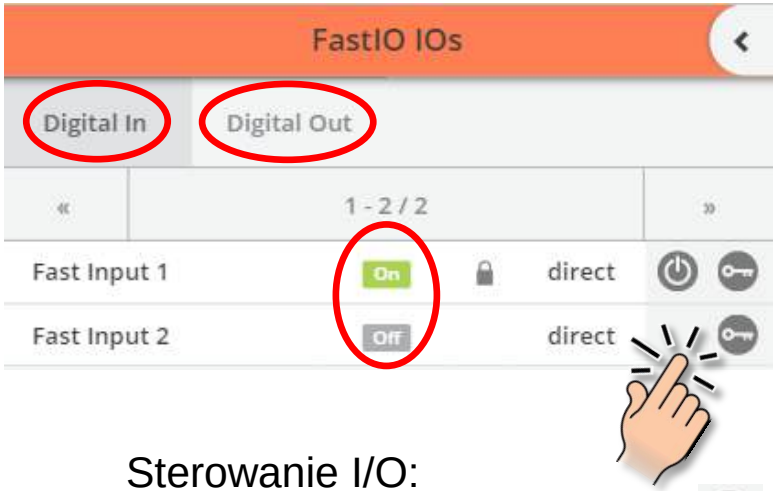
Dostęp do menu I/O z domowej strony menu głównego ->



- W standardzie: 2 szybkie wejścia & 2 wyjścia (J212)







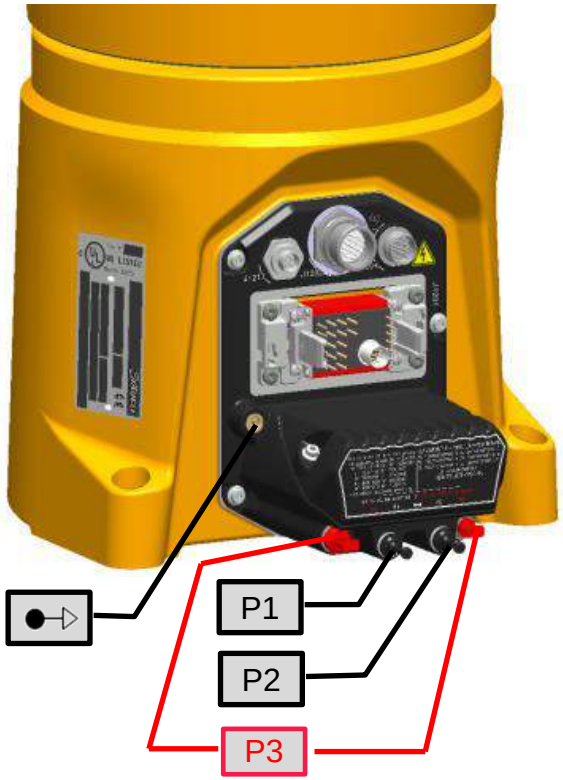
- 1 strona na wejścia & 1 strona na wyjścia
- Status I/O (ON lub OFF)

Sterowanie I/O:

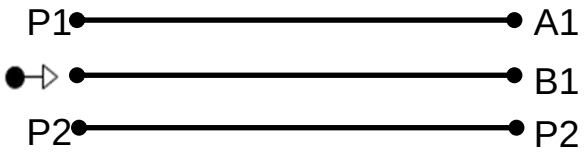
- Załączenie lub wyłączenie (tylko wyjścia)
- Zablokowanie

WYJŚCIA CYFROWE sterowanie zaworami

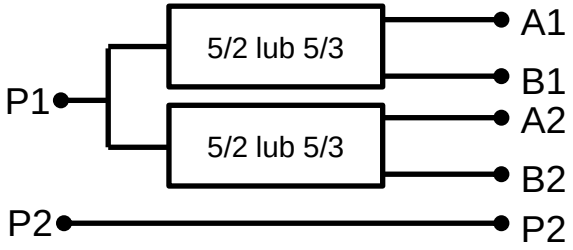
- Rodzaje zaworów (w standardzie 2 zawory z wyjątkiem TX2-40 – 1 zawór)



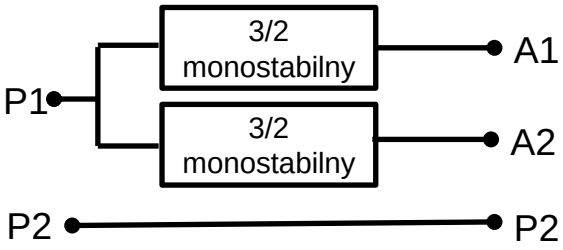
STANDARD: 3 przewody bezpośrednie



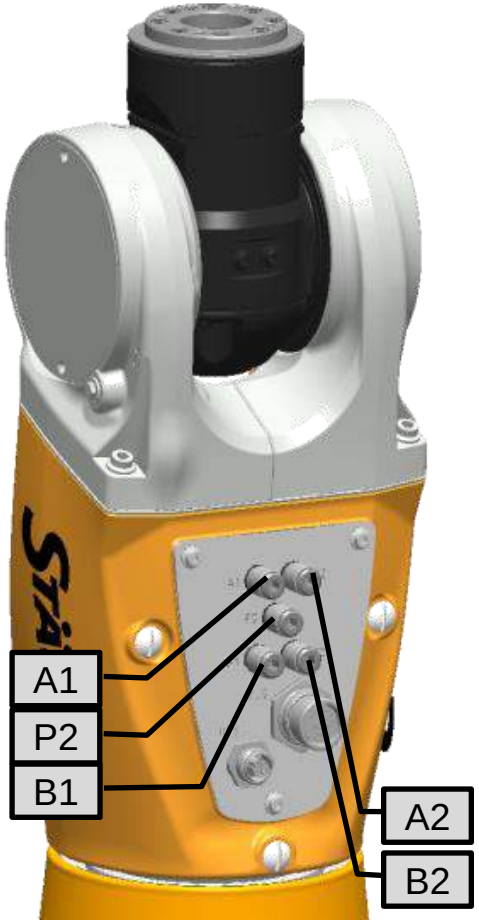
Opcja 1: 2 zawory bistabilne 5/2 (ciśnienie)
Lub 2 zawory monostabilne 5/3 (ciśnienie)



Opcja 2: 2 zawory monostaiblne 3/2 (próżnia)



P3: **zwiększenie ciśnienia w korpusie**

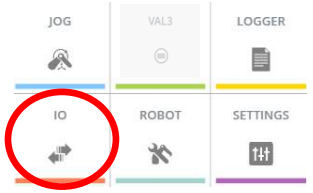


Przykład TX2-90

WYJŚCIA CYFROWE sterowanie zaworami

- Zasilanie zaworów z karty DSI (safety)

(karta DSI znajduje się z tyłu podstawy ramienia pod przyłączem przewodu zasilającego)



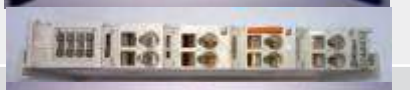
Dostęp do menu I/O z domowej strony menu głównego

- Wyjścia sterujące zaworami (ON lub OFF)

Sterowanie I/O:

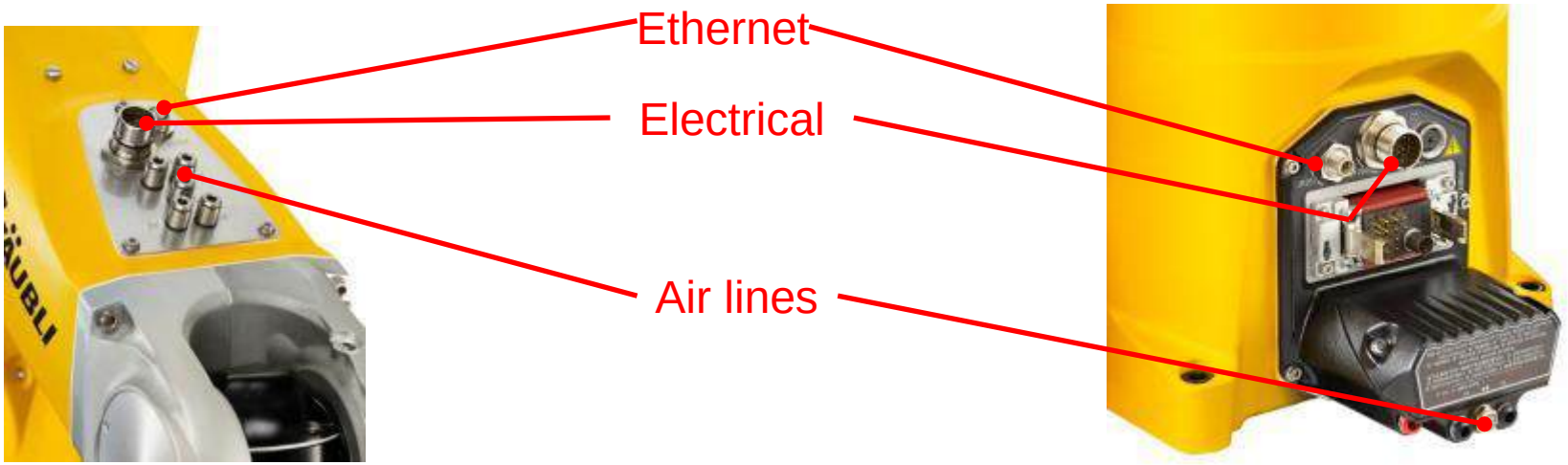
- Załączenie lub wyłączenie (tylko wyjścia)
- Zablokowanie

MODUŁY I/O IP20, (MONTOWANE NA SZYNIE DIN ETHERCAT MASTER)

Opis	Numer		Zużycie prądu
EtherCAT coupler 8 Digital IN + 4 Digital OUT	D244 030 xx		Zasilanie 24V 1A
8 Dgital IN + 8 Digital OUT	D244 031 xx		130 mA
4 Analogic Inputs -10 .. +10V, 16 bits	D244 427 xx		180 mA
4 Analogic Outputs -10 .. +10V, 16 bits	D244 428 xx		265 mA
1 x encoder input ABZ, 5VDC	D244 429 xx		130 mA
2 x RS232	D244 430 xx		130 mA
2 x RS422/RS485	D244 431 xx		270 mA
1 Ebus Power Supply 24V-2A	D244 432 xx		Supply 24V 2A
1 bus end cap	D244 032 xx		

TX2: PODŁĄCZENIE NARZĘDZIA

- **Standard**
  - Przewód Ethernet (cat 5e) przeprowadzony przez ramię robota
  - Wiązka przewodów użytkownika przeprowadzona przez ramię robota
  - Przewody pneumatyczna przeprowadzona przez ramię robota
  - Przyłącza na podstawie i przed nadgarstkiem
  - Złącza umożliwiające zwiększenie ciśnienia w ramieniu
- **Opcje**
  - Wbudowane elektrozawory
  - Moduły EtherCAT I/O instalowane na przedramieniu robota



MODUŁY IO IP67 (MONTOWANE NA RAMIENIU LUB CHWYTAKU)

	Opis	Numer	
Moduł EtherCAT	TX2-90 8 DI or DO Zestaw z mocowaniem i kablami	D244 464 xx	
EtherCAT box	8 DI or DO	D244 435 xx	
Kable	Power cable 1m	D244 436 xx	
	EtherCAT cable 1m	D244 437 xx	
Konektory	I/O connector TX2-40 (J1202/J1203, J1217/J1218)	D244 468 xx	
	I/O connector TX2-60 (J1202/J1203, J1217/J1218)	D244 467 xx	
	I/O connector TX2-90 (J1202/J1203, J1217/J1218)	D244 463 xx	

- Dokumentacja dostępna pod adresem: http://www.staubli.com/en/robotics/robot-controller/robot-controller-cs9/io/

ODCZYT STANU WEJŚĆ

- Sprawdzanie stanu wejść

  diCapteur == true — zwraca 1 jeżeli aktywne

  diCapteur == false — zwraca 1 jeżeli nieaktywne

Możliwe wykorzystanie w poleceniach:

- Warunkowe:

```
if (diSensor == true)
….
else
….
endIf
```

- Pętle:

```
do
….
until (diSensor == true)
```

```
While (diSensor == true)
….
endWhile
```

- Oczekiwanie na sygnał: wait(diSensor == true)

ODCZYT STANU WEJŚĆ – c.d.

Oczekiwanie na sygnał:

- **wait(iSensor==true)**

Oczekiwanie przez określony czas na sygnał:

- **watch(iSensor==true,2)**
- zwraca **true** jeżeli warunek będzie spełniony w przeciągu maximum 2 sekund, w przeciwnym wypadku zwraca **false**

```
Zadeklarowano:
point pA
tool tSucker
mdesc mFast
dio iSensor

program main()
begin
  movej(pA, tSucker, mFast)
  if watch(iSensor == true, 5) == false
    putln("Failure from Input iSensor")
    wait(iSensor == true and iQuit == true)
  endIf
end
```

STEROWANIE WYJŚCIAMI

- doConveyor=true — aktywacja wyjścia
- doConveyor=false — dezaktywacja wyjścia

SYNCHRONIZACJA RUCHÓW I STEROWANIA WE/WY

VAL3 : Aby zsynchronizować sterowanie wejściami/wyjściami z ruchami robota:

waitEndMove()

np.

movej(pControl,tGrip, mFast)

waitEndMove()

DoStartcontrol=true

….

Należy wyłączyć blending na poprzedzającym ruchu

INSTRUKCJE WAITENDMOVE ORAZ DELAY

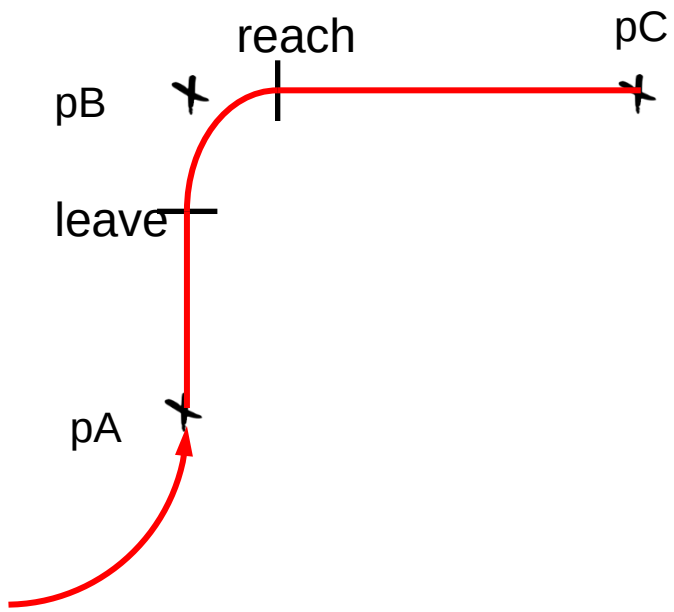
delay(num)

Wstrzymuje wykonywanie programu na podany czas w sekundach

movej(pControl, tGrip, mFast)
waitEndMove()
io:sigmeasure= true
delay(2.5)
io:sigmeasure=false
….

WAIT() – PRZYKŁAD

W którym punkcie robot będzie oczekiwał na sygnał „*iSensor*“ ?




```
Declaration:
tool tSucker
mDesc mFast
point pA, pB, pC,


program main()
begin
  mFast.blend=joint
  mFast.leave=mFast.reach=20
  movej(pA, tSucker, mFast)
  movel(pB, tSucker, mFast)
  wait(iSensor == true)
  movel(pC, tSucker, mFast)
  waitEndMove()
end
```

INSTRUKCJA: DELAY()

delay(num) zatrzymuje wykonywanie programu na określony czas w sekundach.

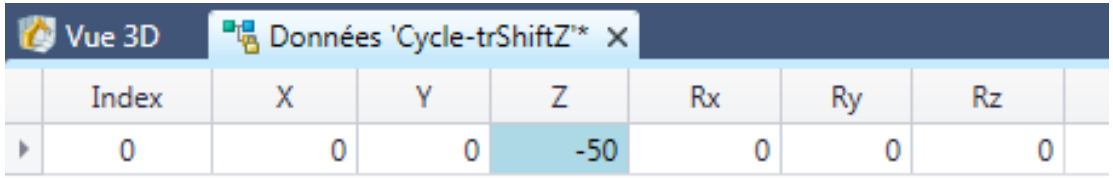
```
Zadeklarowano:
point pControl
tool tSucker
mdesc mFast
dio oOutput

program main()
begin
  movej(pControl, tSucker, mFast)
  waitEndMove()
  io:oOutput = true
  delay(2.5)
  io:oOutput = false
end
```

Załącza wyjścia ***oOutput*** na 2,5 sekundy

- CZĘŚĆ 2 – USER CS9

OBLICZANIE POZYCJI DOJAZDU DO PUNKTÓW KARTEZJAŃSKICH

TYP ZMIENNEJ TRANSFORMATY: TRSF



Index	X	Y	Z	Rx	Ry	Rz
0	0	0	-50	0	0	0

*Zmienna transformaty **TRSF** pozwala na przeliczanie współrzędnych punktów kartezjańskich*

Np: Dojazd do punktu, obliczenie współrzędnych na palecie, ….

6 zmiennych numerycznych: x, y, z, rx, ry, rz

Definicja transformaty: trShift

W programie :

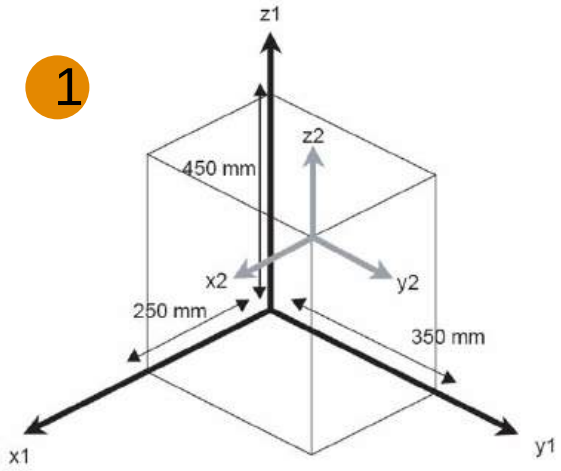
trShift={0,0,-100,0,0,0}

lub trShift.x=0 trShift.y=0 trShift.z=-100 trShift.rx=0 ...

Nie ma możliwości wykonywania ruchów bezpośrednio do transformat !!! Używane tylko z punktami kartezjańskimi !!!

TRANSFORMATA {X, Y, Z, RX, RY, RZ} = {250,350,450,20,10,30}

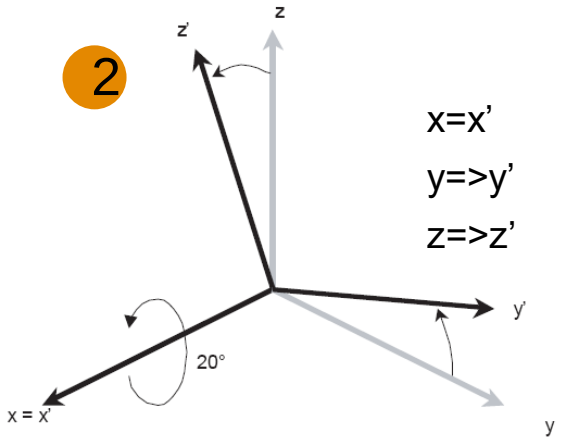
1



{250,350,450,0,0,0}

2

RX



x=x'

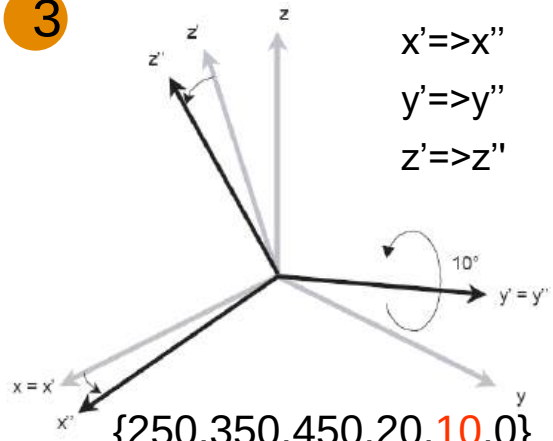
y=>y'

z=>z'

{250,350,450,20,0,0}

3

RY



x'=>x''

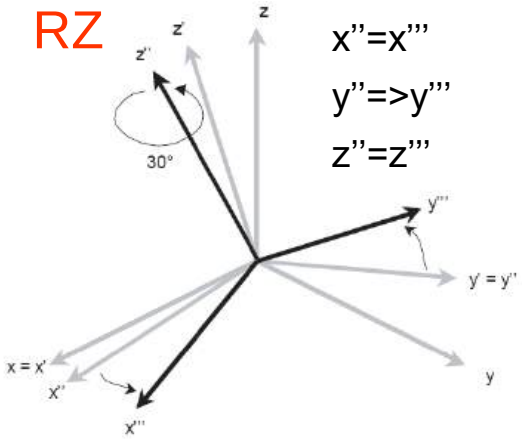
y'=>y''

z'=>z''

{250,350,450,20,10,0}

4

RZ



x''=x'''

y''=>y'''

z''=z'''

{250,350,450,20,10,30}

APPRO na punktach kartezjańskich POINT

POINT <== appro(POINT,TRSF)

APPRO oblicza współrzędne punktu kartezjańskiego przesuniętego względem bazowego punktu kartezjańskiego z wykorzystaniem transformaty

Zdefiniowano: POINT **pTemp** POINT **pPick** TRSF **trShiftz** NUM **nDistance**=100

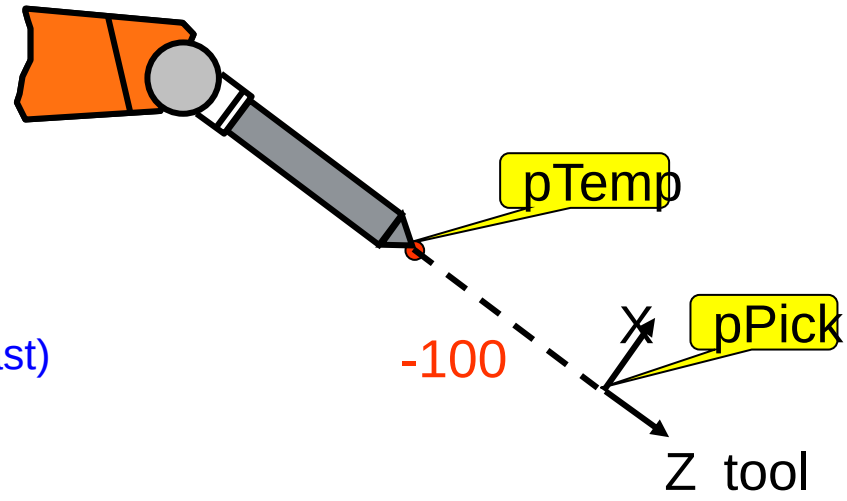
```
trShiftz={0,0,-nDistance,0,0,0}
pTemp=appro(pPick,trShiftz)
movej(pTemp,tGrip,mFast)
```

Inny zapis:

```
movej(appro(pPick,trShiftz),tGrip,mFast)
```

Inny zapis:

```
movej(appro(pPick,{0,0,-100,0,0,0}),tGrip,mFast)
```



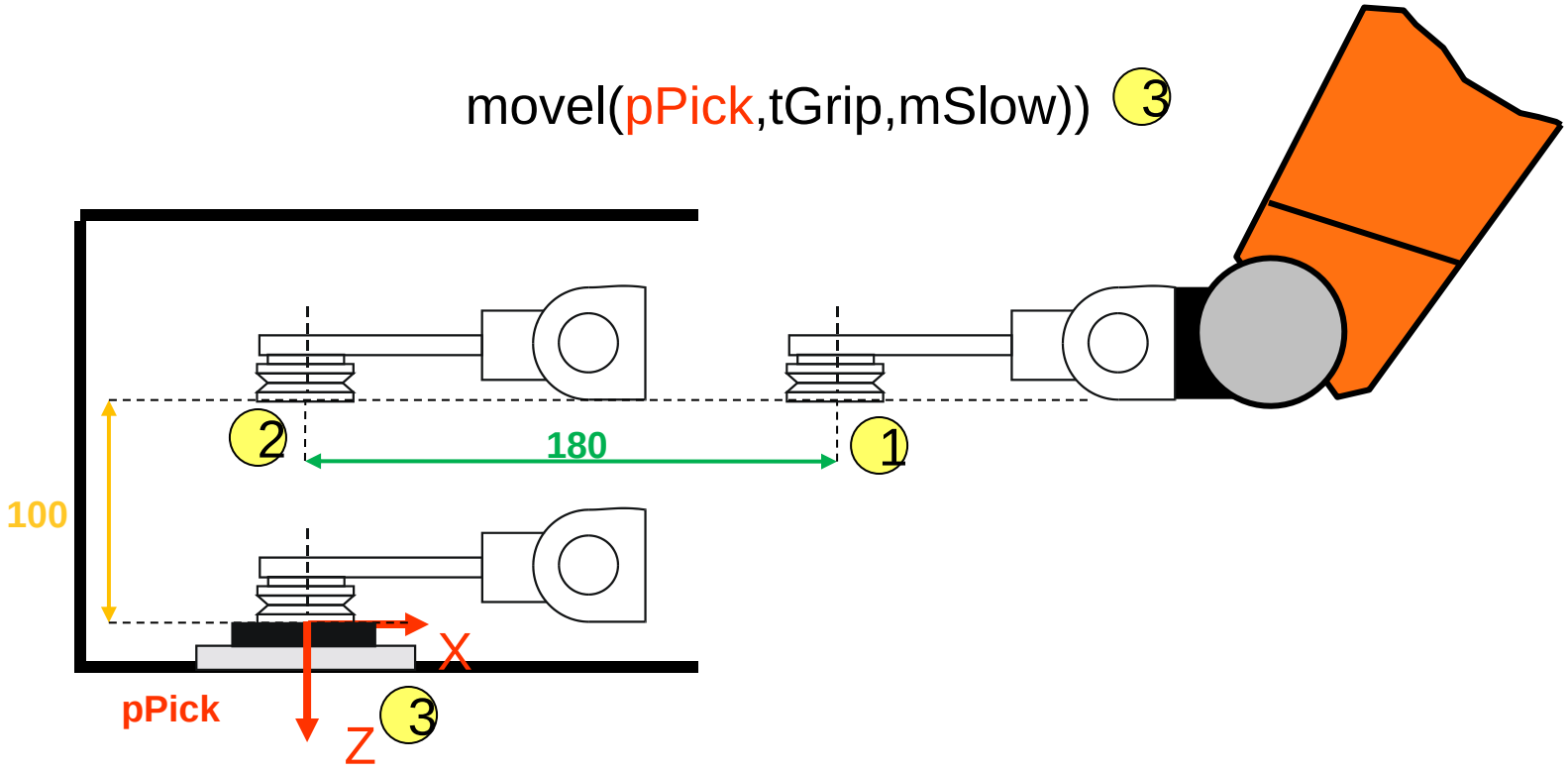
appro związane jest z układem narzędzia – dojazd do punktu

ZŁOŻONY DOJAZD DO PUNKTU – C.D.

movej(appro(pPick,{180,0,-100,0,0,0}),tGrip,mSlow) 1

movel(appro(pPick,{0,0,-100,0,0,0}),tGrip,mSlow) 2

movel(pPick,tGrip,mSlow)) 3

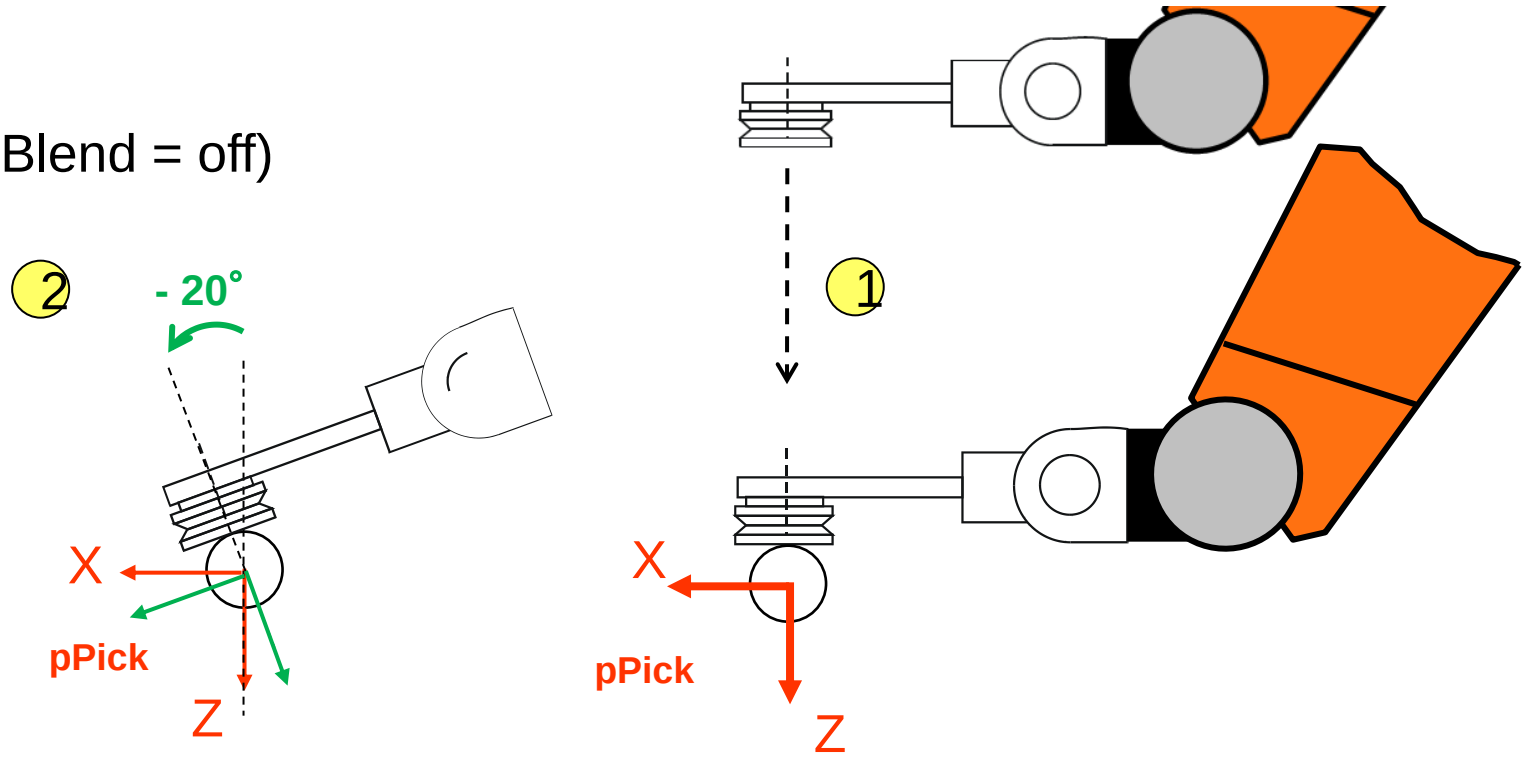


ZŁOŻONY DOJAZD DO PUNKTU – C.D.

movel(pPick,tGrip,mSlow) 1

movel(appro(pPick,{0,0,0,0,**-20**,0}),tGrip,mSlow) 2

(Blend = off)



WYKORZYSTANIE FUNKCJI APPRO – ćwiczenie

Ćwiczenie nr 8: „Zamiana miejsc” – zadanie 3

1. Proszę napisać program do zamiany miejsc, zgodnie z poleceniami w zadaniu 3, wykorzystując do tego celu program SRS.
2. Proszę obliczyć pozycje punktów pośrednich wykorzystując funkcję **APPRO**.

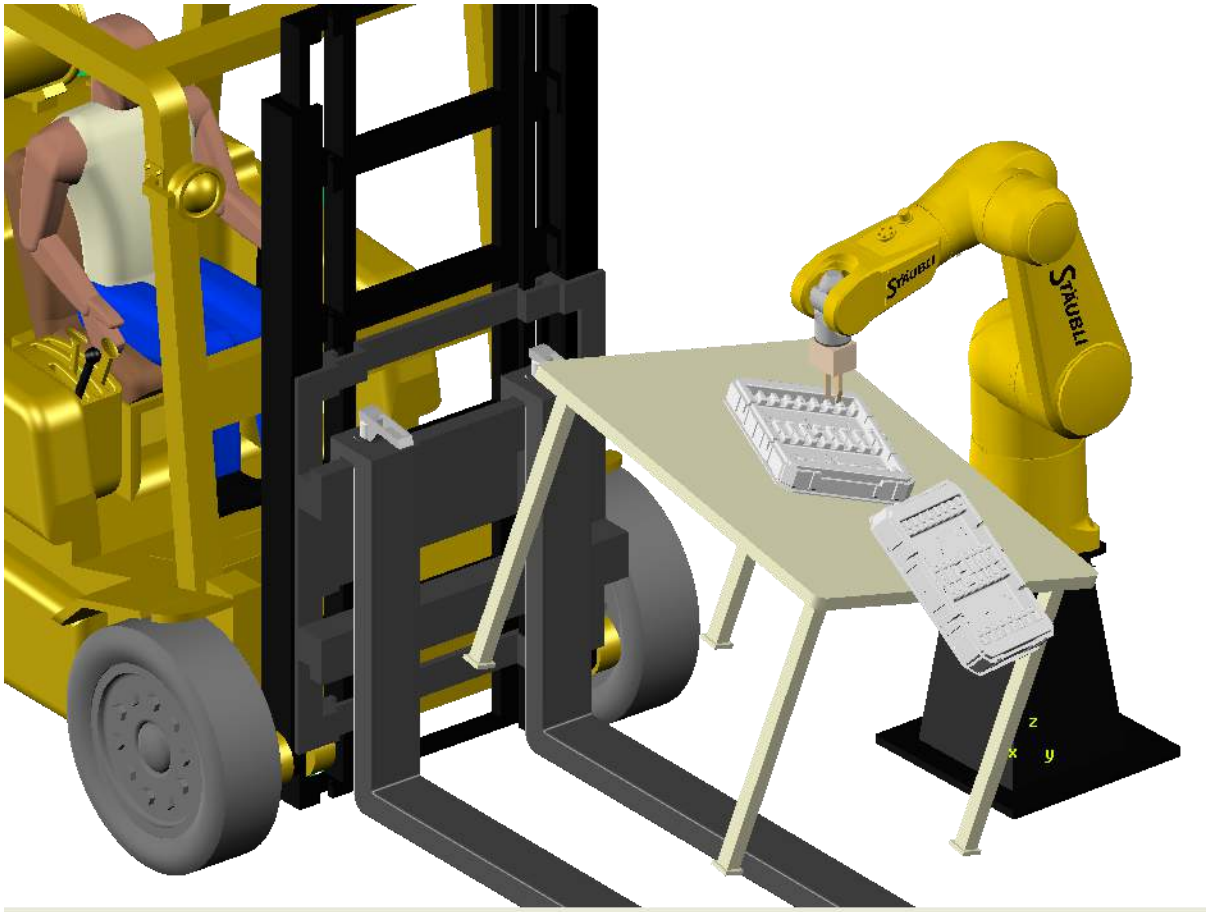
CZĘŚĆ 2 – USER CS9

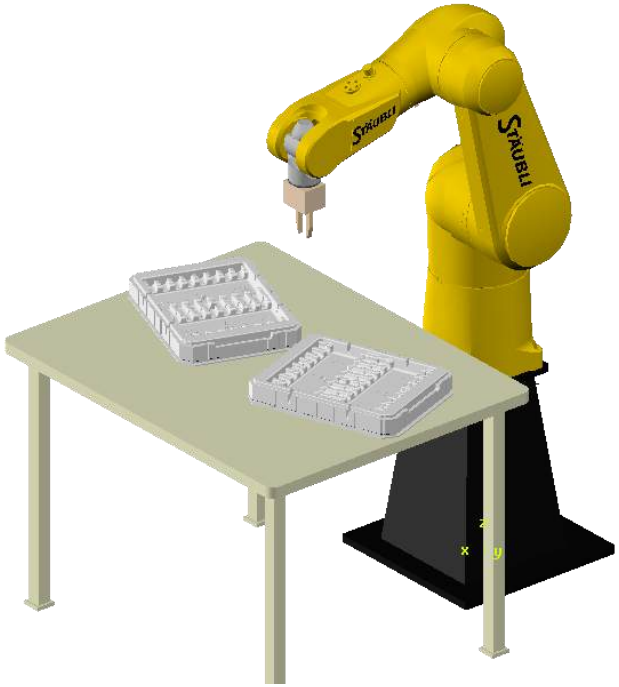
UKŁADY WSPÓŁRZĘDNYCH

DO CZEGO WYKORZYSTUJEMY UKŁADY WSPÓŁRZĘDNYCH?

Robot pracuje na linii

Aplikacja działa z pełną prędkością …





Janek jedzie wózkiem widłowym ...

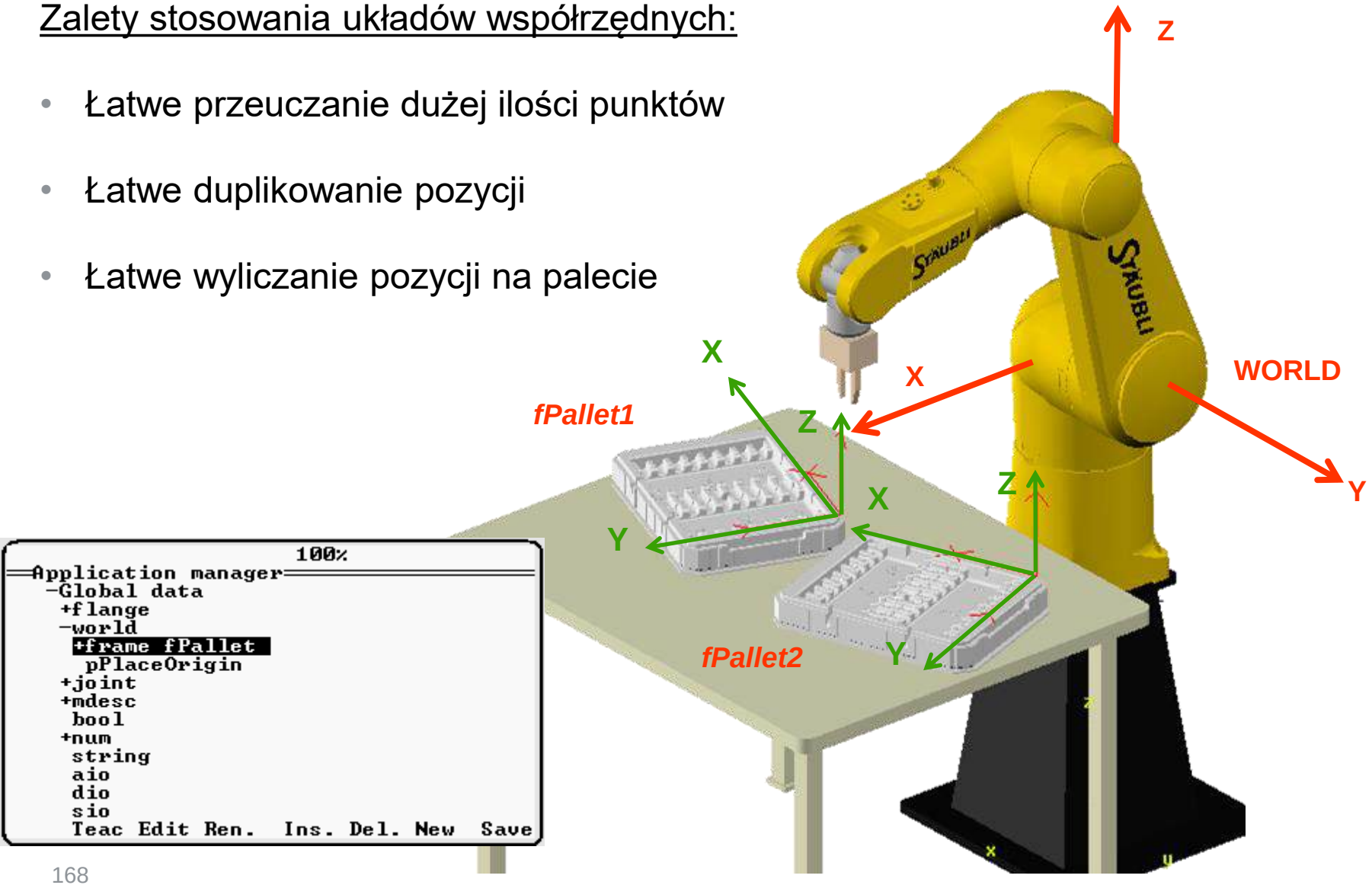
!!!! KATASTROFA !!!!!

… jeden dzień na przeuczanie punktów …

DO CZEGO WYKORZYSTUJEMY UKŁADY WSPÓŁRZĘDNYCH?

Zalety stosowania układów współrzędnych:

- Łatwe przeuczanie dużej ilości punktów
- Łatwe duplikowanie pozycji
- Łatwe wyliczanie pozycji na palecie

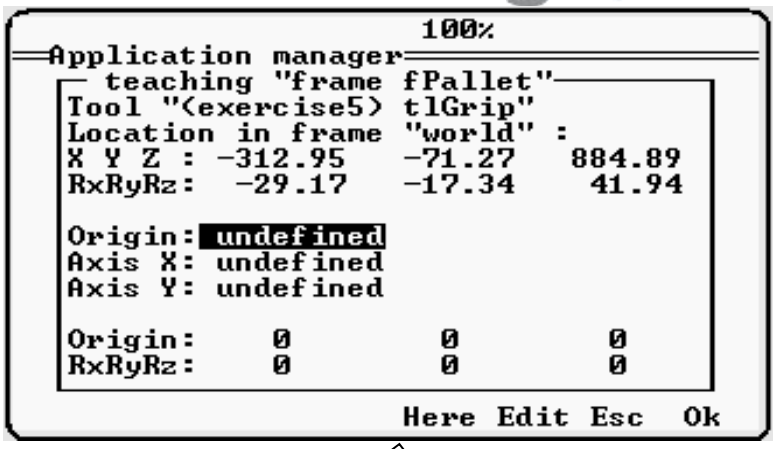


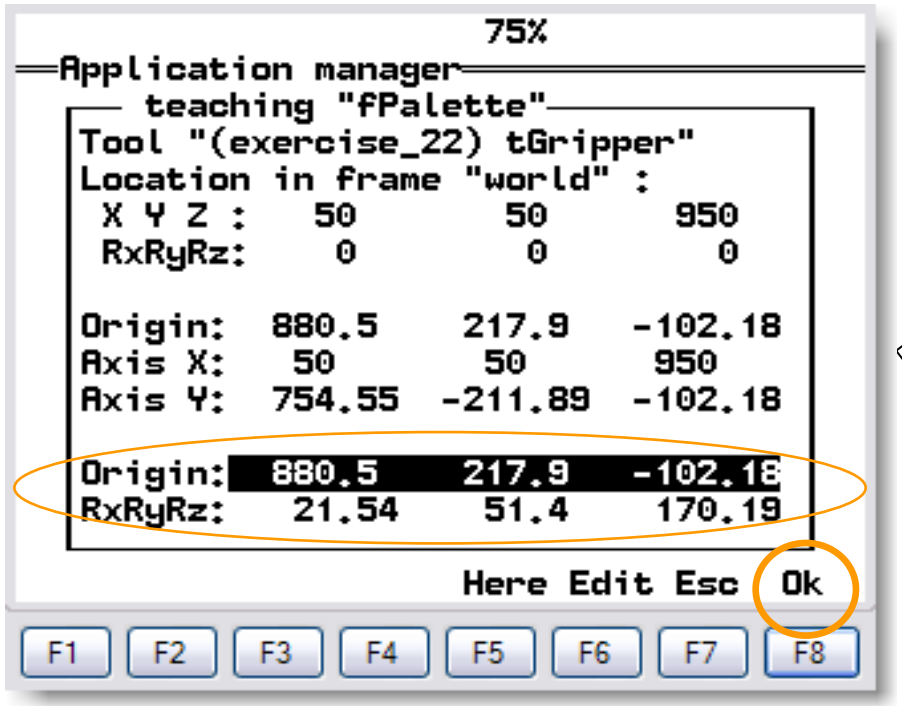
NAUKA UKŁADU WSPÓŁRZĘDNYCH

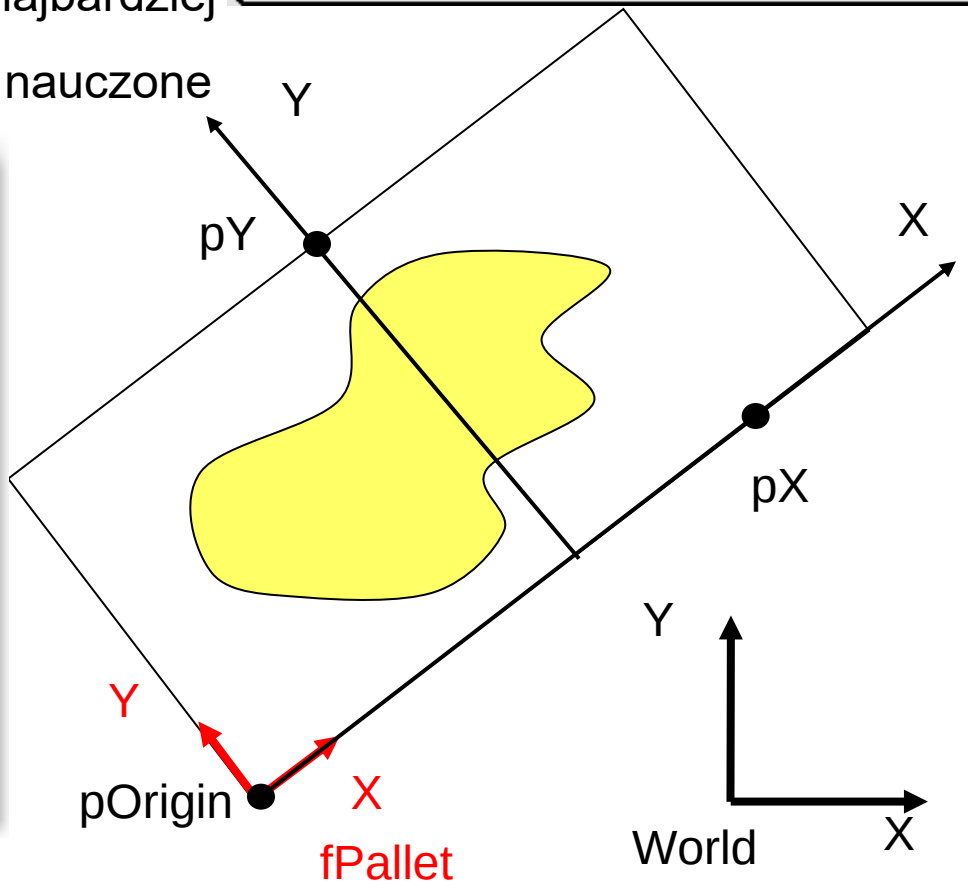
STÄUBLI

Definicja przez naukę 3 punktów:

- Wykorzystaj precyzyjne narzędzie
- Ustaw je jako aktywne narzędzie
- Punkty referencyjne powinny być jak najbardziej oddalone od siebie i jak najdokładniej nauczone





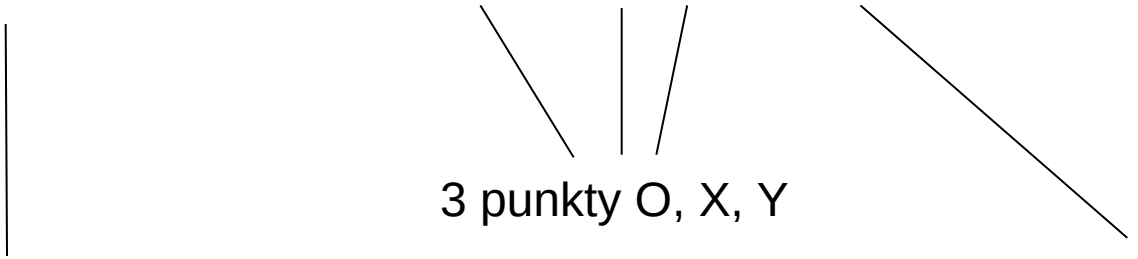


NAUKA UKŁADU WSPÓŁRZĘDNYCH – C.D.

Nauka układu współrzędnych to prosta wyznaczana przez 2 punkty **Origin** i **X**. **Y** uczymy gdzie chcemy zgodnie z kierunkiem wyznaczonym z reguły prawej dłoni (układ współrzędnych prawoskrętny). System wyznaczy prostą prostopadłą przebiegającą przez punkt Y do prostej pomiędzy punktami Origin i X (zostanie wyznaczona płaszczyzna).

OBLICZANIE UKŁADU WSPÓŁRZĘDNYCH PRZEZ PROGRAM

nError = setFrame(pOrigin, pX,pY, fPallet)



Kod błędu:

0 : układ obliczony prawidłowo

-1 : pX za blisko pOrigin

-2 : 3 punkty są w pobliżu jednej prostej

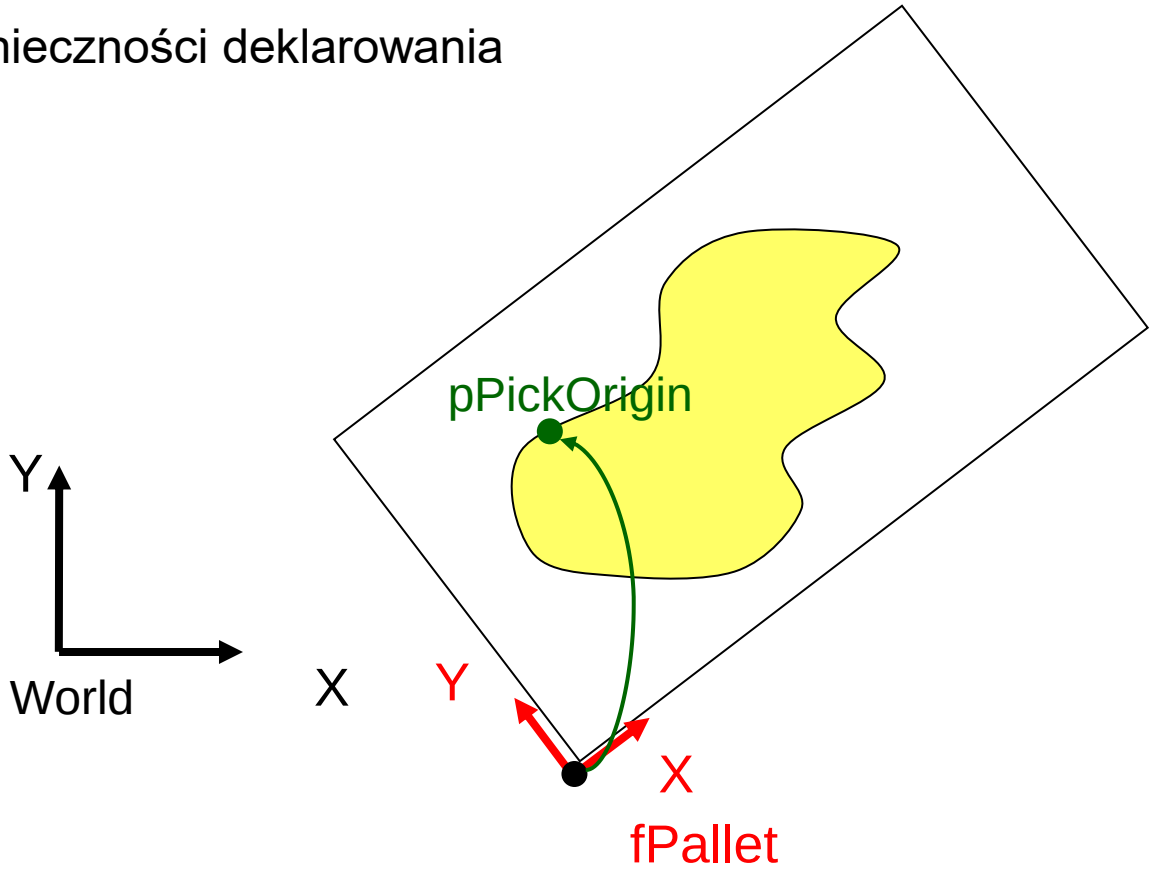
Nazwa wyliczonego układu współrzędnych

PUNKTY W UKŁADZIE WSPÓŁRZĘDNYCH

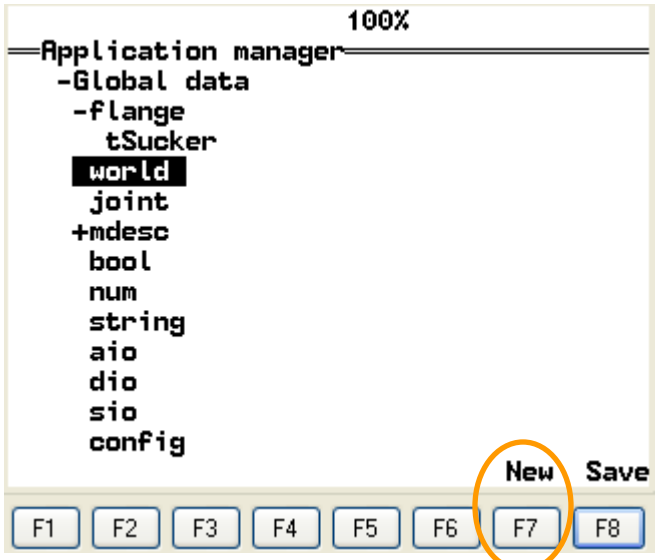
Punkty tworzymy w drzewku układu współrzędnych:

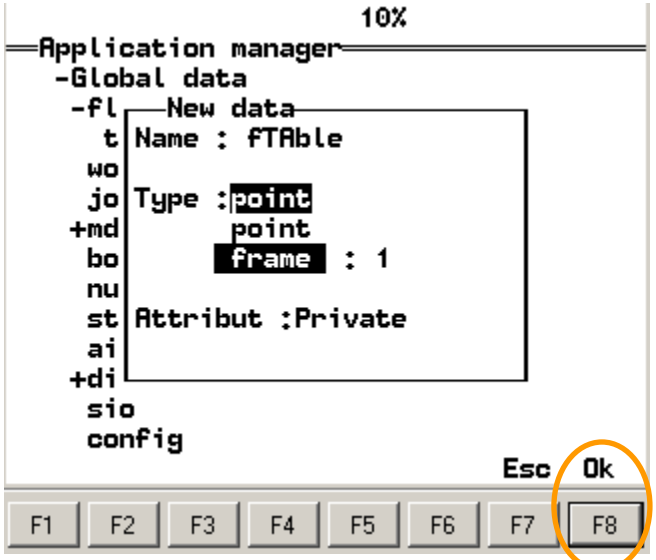
- Podczas uczenia, współrzędne są wyliczane względem danego układu współrzędnych
- W instrukcji ruchu nie ma konieczności deklarowania układu współrzędnych:

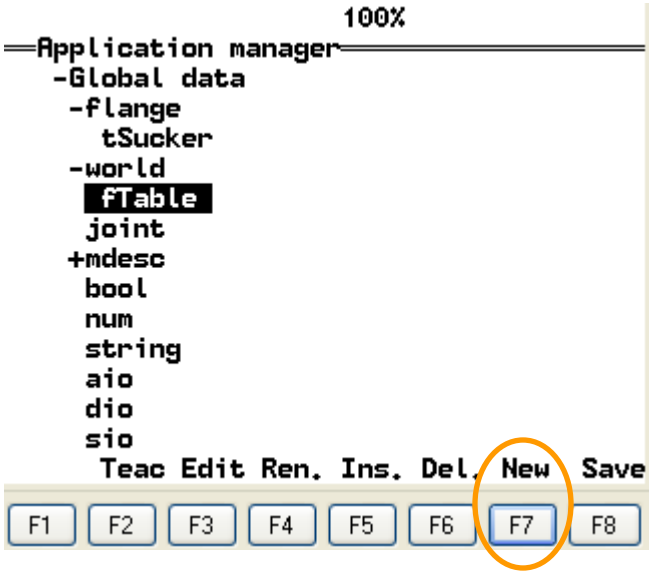
movej(pPickOrigin,tGrip,mFast)

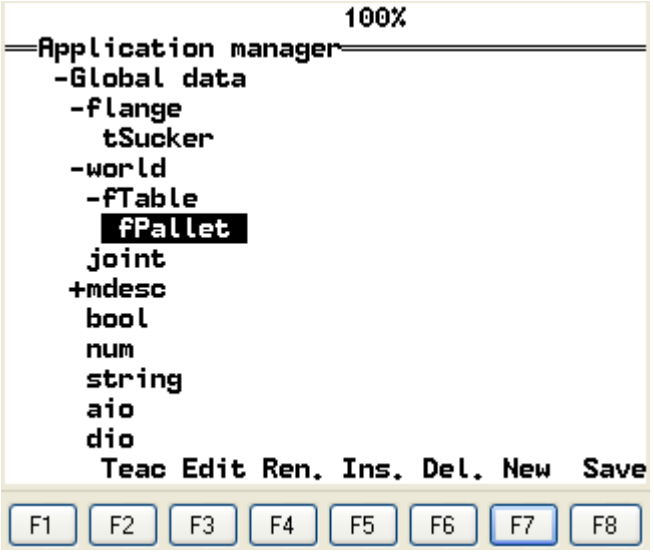


TWORZENIE ZMIENNEJ TYPU FRAME







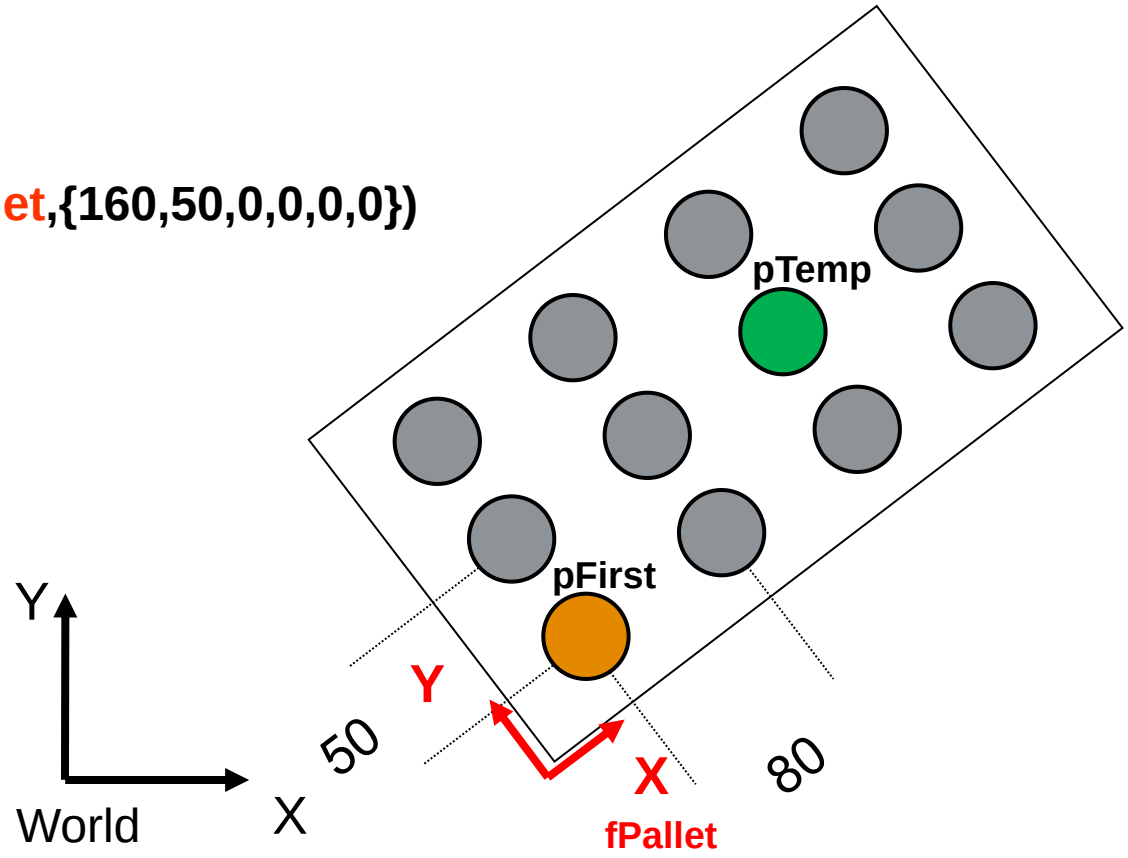


PALETYZACJA W UKŁADZIE WSPÓŁRZĘDNYCH

Compose(point,frame,trsf): wylicza punkt przesunięty o zadaną transformatę względem układu współrzędnych np. palety (związany jest z układem współrzędnym)

pTemp=compose(pFirst,fPallet,{160,50,0,0,0,0})

movel(pTemp,tGrip,mSlow)

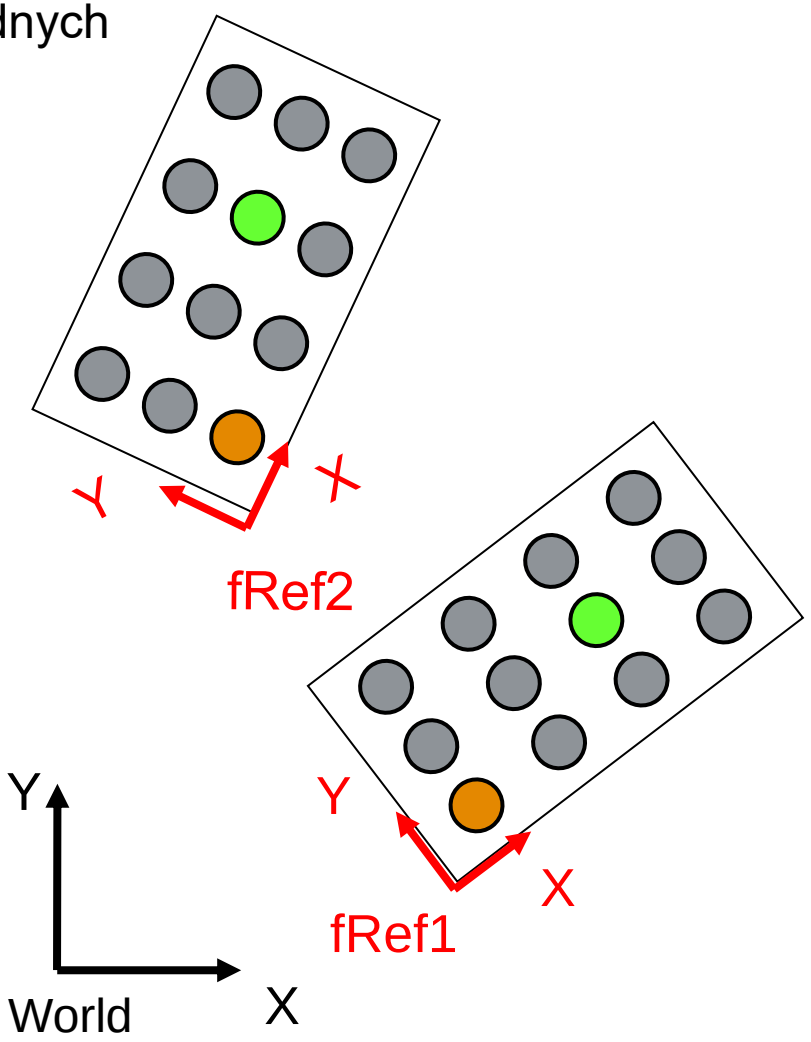


WYKORZYSTANIE DWÓCH IDENTYCZNYCH PALET

Aby wykorzystać ten sam punkt w dwóch różnych układach współrzędnych:

- Zdefiniuj punkty w obydwu układach współrzędnych
- Naucz jeden z punktów
- Skopiuj **trsf** jednego punktu do drugiego

pRef2.trsf=pRef1.trsf



UKŁADY WSPÓŁRZĘDNYCH I PALETYZACJA – ćwiczenie

Ćwiczenie nr 9: Nauka układów współrzędnych

1. Proszę nauczyć 2 układy współrzędnych (płaski i skośny), następnie utworzyć w nich punkty.
2. Proszę wykorzystać odpowiednie narzędzia (do nauki układu współrzędnych pointer i chwytak).
3. Proszę włączyć odpowiedni tryb poruszania robotem i poruszać ramieniem wzdłuż osi współrzędnych zaprogramowanych układów
4. Proszę ustawić narzędzie prostopadle do płaszczyzny zaprogramowanych układów współrzędnych.

SZKOLENIE Z OBSŁUGI ROBOTÓW STÄUBLI

cz.3 – poziom programista

Seria robotów: TX2/**CS9**

TABELA ZAGADNIEŃ – PROGRAMISTA

- CZĘŚĆ 3 – PROGRAMISTA CS9

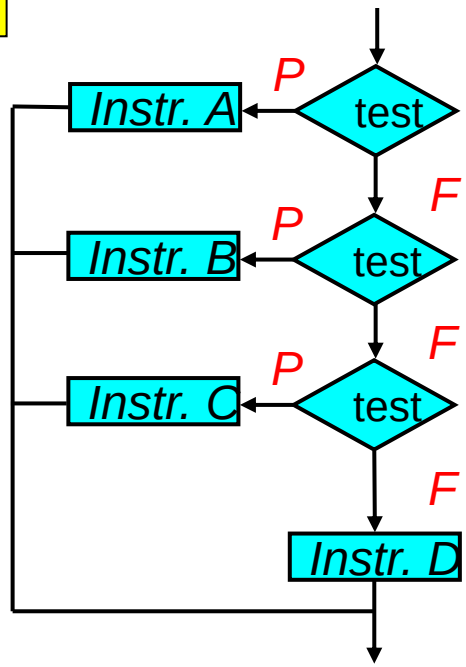
PROGRAMOWANIE STRUKTURALNE

OPERATORY – WARUNKI: IF

Operatory: **<**, **>**, **==**, **!=**, **>=**, **<=**, **and**, **or**, **xor**, **!**

Warunki: nX == 1 (true) lub (false) nY != 3 nY nie równe 3

```
If nX==1
Instrukcje VAL3
endIf
```




```
if nY != 3
                | Instrukcje A
elseIf nX>2
                | Instrukcje B
elseIf (nY==5) or (nX==6)
                | Instrukcje C
else
                | Instrukcje D
endIf
```

PRZYKŁADY: IF

Przykład 1

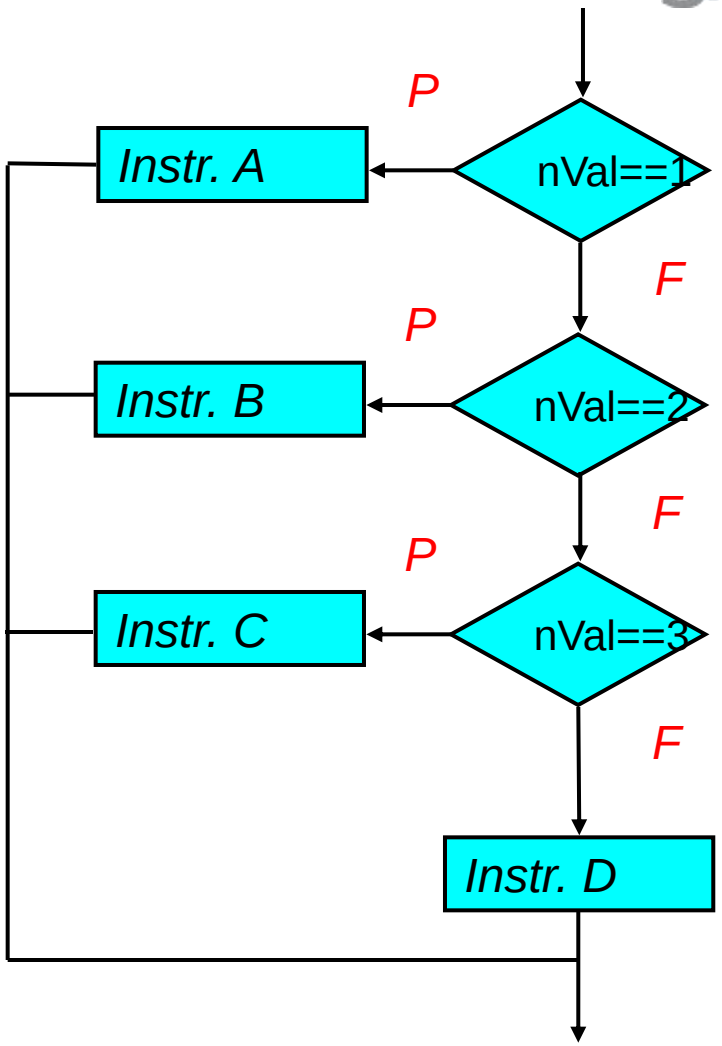
```
Zadeklarowano:
num nTemperature
string sCommand
bool bEnd

if nTemperature>20
    | Block A
else
  if sCommand == “Radiator On“
    | Block B
  else
    if bEnd != true
    | Block C
    endIf
  endIf
endIf
```

Przykład 2

```
Zadeklarowano:
num nTemperature
string sCommand
bool bEnd

if nTemperature>20
    | Block A
elseIf sBefehl == “Radiator On“
    | Block B
elseIf bEnd != true
    | Block C
endIf
```

POLECENIE: switch

```
switch nVal
case 1
        | Instrukcje . A
break
case 2
        | Instrukcje . B
break
case 3
        | Instrukcje . C
break
default
        | Instrukcje . D
break
endSwitch
```



CASE 1,2,3: **nVal** == **1** lub **2** lub **3**

PRZYKŁADY – SWITCH

Przykład 1

```
Zadeklarowano:
num nTemperature
string sCommand
bool bEnd

switch true
   case nTemperature>20
      | Block A
   break
   case sBefehl == "Radiator On"
      | Block B
   break
   case bEnd != true
      | Block C
   break
endSwitch
```

Przykład 2

```
Zadeklarowano:
string sText

switch sText
   case "Arm power on"
      | Block A
   break
   case "Arm power off"
      | Block B
   break
   case "new position"
      | Block C
   break
endSwitch
```

PĘTLE – definicja: for

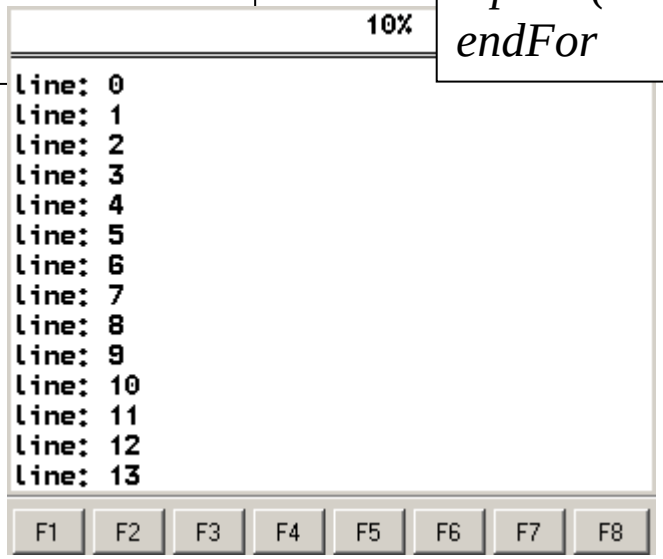
```
for nIndex= start to koniec step inkrementacja
<Instrukcje>
endFor
```

```
for nIndex=0 to 50
        movel(pTraj[nIndex],tGrip,mFast)
endFor
```

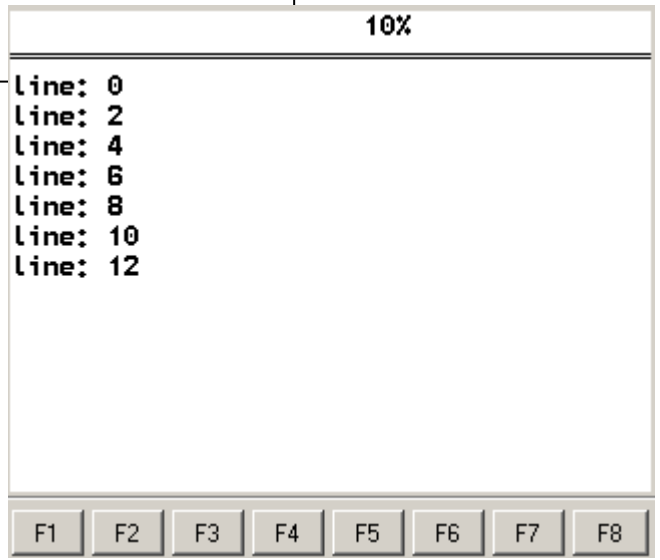
```
for nIndex = 100 to 10 step -1          step <> 1
        nShift[nIndex]= nH*nIndex
endFor
```

PĘTLE: DEFINIOWANIE ILOŚCI PĘTLI

```
Declaration:
num nIndex

for nIndex = 0 to 13
   gotoxy(0,nIndex)
   put(“line: “)
   put(nIndex)
endFor
```




```
Declaration:
num nIndex
string sMessage

for nIndex = 0 to 13 step 2
    put(“line: “)
   putln(nIndex)
endFor
```

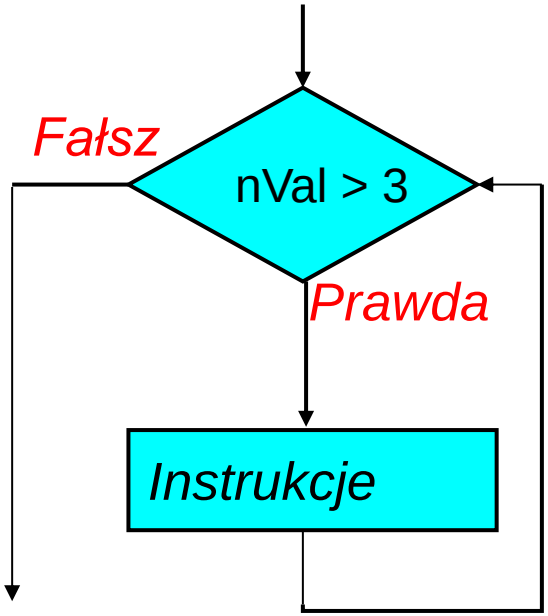


185

PĘTLE WARUNKOWE: WHILE ORAZ DO

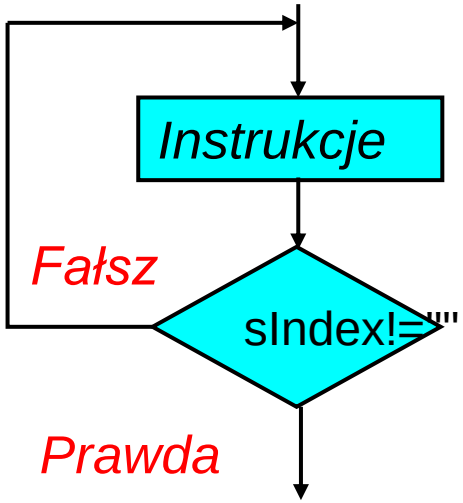
STÄUBLI

```
while nVal > 3
    |
    |  Instrukcje
endWhile
```




```
while get() != 271
    putln("Wrong key pressed")
endWhile
putln("The F1 key was pressed")
```

```
do
    |
    | Instrukcje
until sIndex !=" "
```

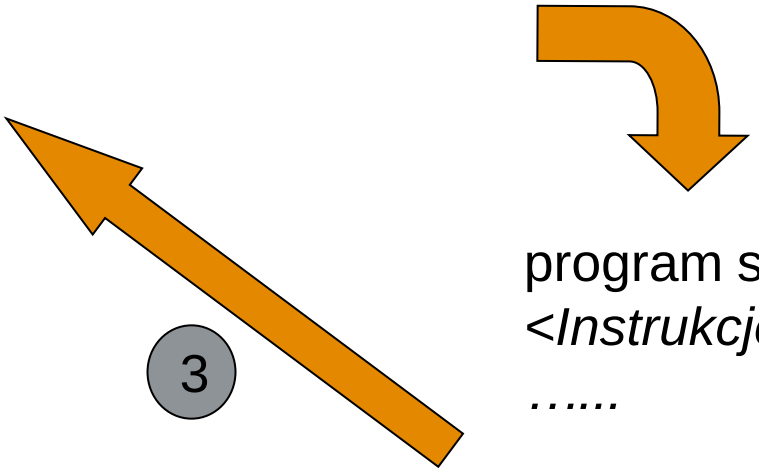



```
Declaration:
num nIndex

do
   nIndex=get()
until nIndex==271
putln("The F1 key was pressed")
```

PODPROGRAMY

```
program appli()
....
....
CALL subprog()
....
....
```

1



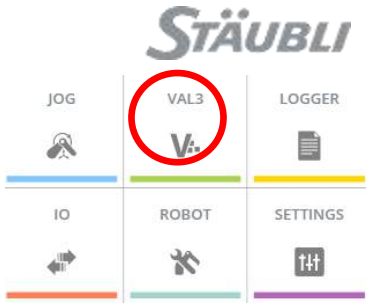
```
program subprog()
<Instrukcje>
.....
```

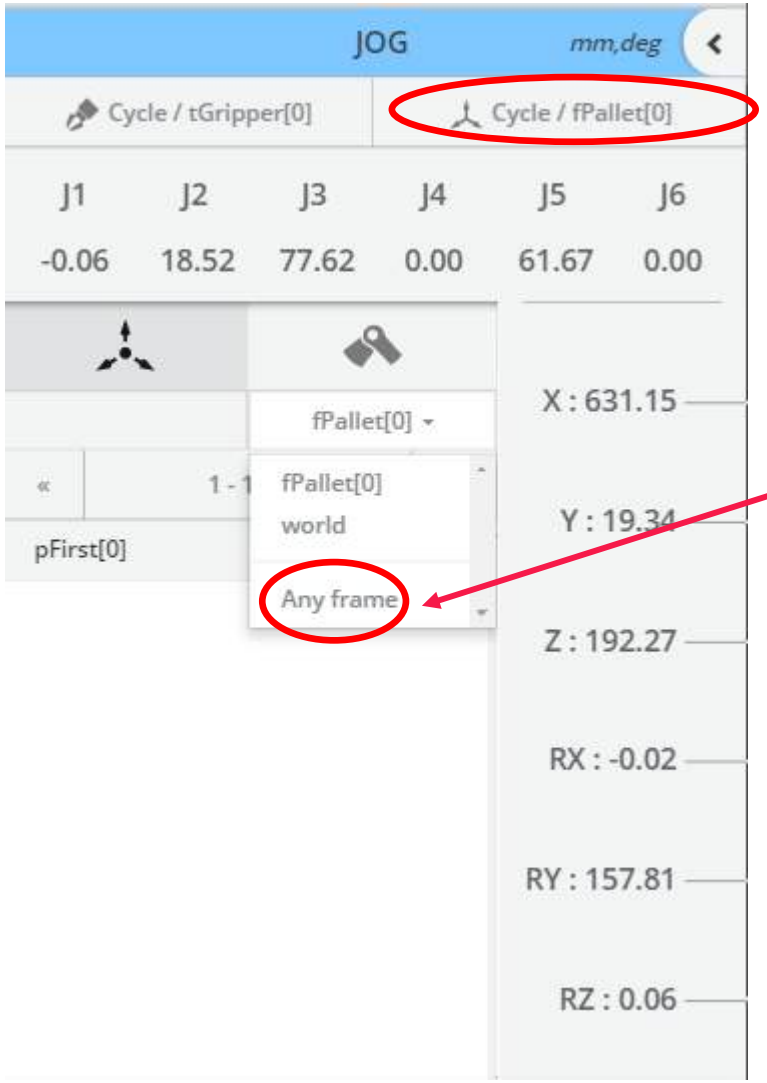
2

3

3 Powrót do programu appli z podprogramu subprog
Możliwość wykorzystania instrukcji **RETURN**

JOGGING MODE & local Frame





Bieżący układ współrzędnych używany do poruszania się robota

Filtr wyświetlania punktów:

- **« Any frame »** wyświetla wszystkie punkty ze wszystkich układów współrzędnych (również z układu « **World** »)
- **World** wyświetla wszystkie punkty położone względem układu współrzędnych « **World** »
- **fPallet** wyświetla wszystkie punkty powiązane z układem współrzędnym « **Pallet** »

ROBOTICS

STÄUBLI

- CZĘŚĆ 3 – PROGRAMISTA CS9

EDYCJA ZŁOŻONYCH TYPÓW ZMIENNYCH





PARAMETRY MDESC

MDESC	MDESC speed :
.vel (%)	speed.vel =speed.vel/2
.accel (%)	
.decel (%)	
.tvel (mm/s)	
.rvel (deg/s)	
.blending (off / joint)	speed.blending=joint
.leave (mm)	speed.reach=speed.leave= 20
.reach (mm)	
=	
==	
!=	

PARAMETRY POINT

POINT .trsf .config Powiązane z układem współrzędnych (niewidoczne) = != ==	POINT pta : POINT ptb : <u>Kopiowanie współrzędnych punktu do innego punktu</u> : pta=ptb *! ! ! pta przyjmuje taki sam układ współrzędnych jak ptb ! !* <u>Kopiowanie współrzędnych punktu do innego punktu bez układu odniesienia</u>: pta.trsf=ptb.trsf <u>Wpisanie wartości 100 do współrzędnej X punktu pta</u>: pta.trsf.x=100

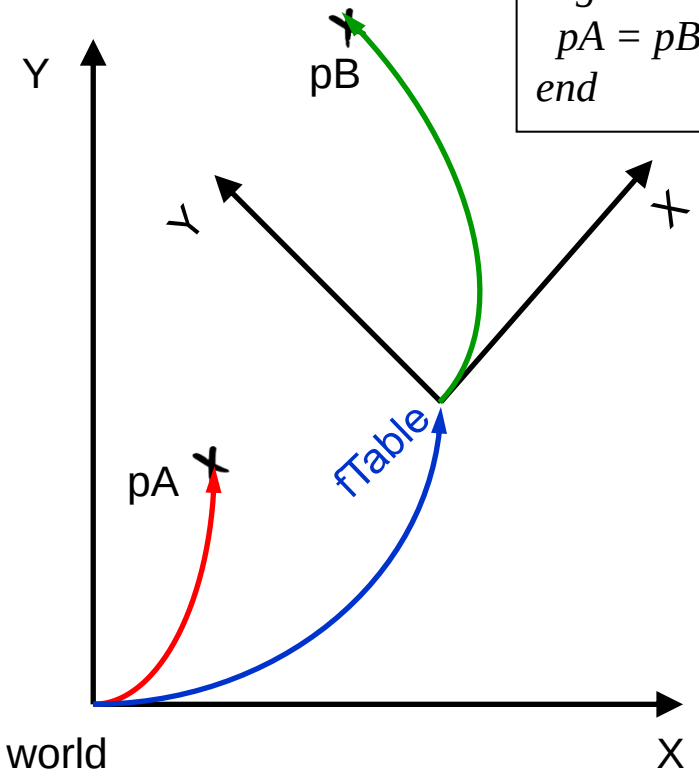
OPERACJE NA ZMIENNYCH POINT

	point: pPoint trsf: trPoint config: cfPoint
= == !=	trPoint = pPoint.trsf cfPoint = pPoint.config pPoint.trsf.x = pPoint.trsf.x + 100

OPERACJE NA ZMIENNYCH POINT – C.D.

Zadeklarowano:
frame fTable
point pA, pB

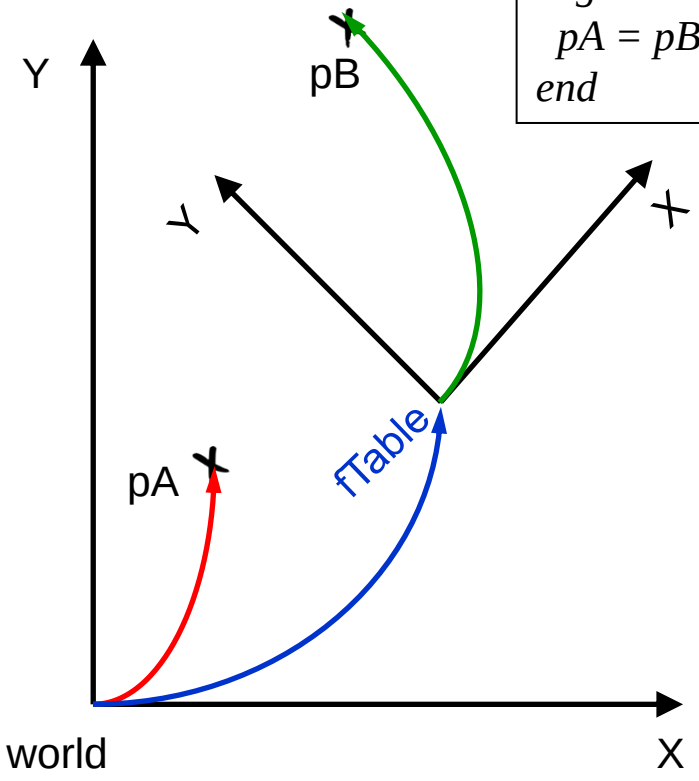
program main()
begin
pA = pB
end

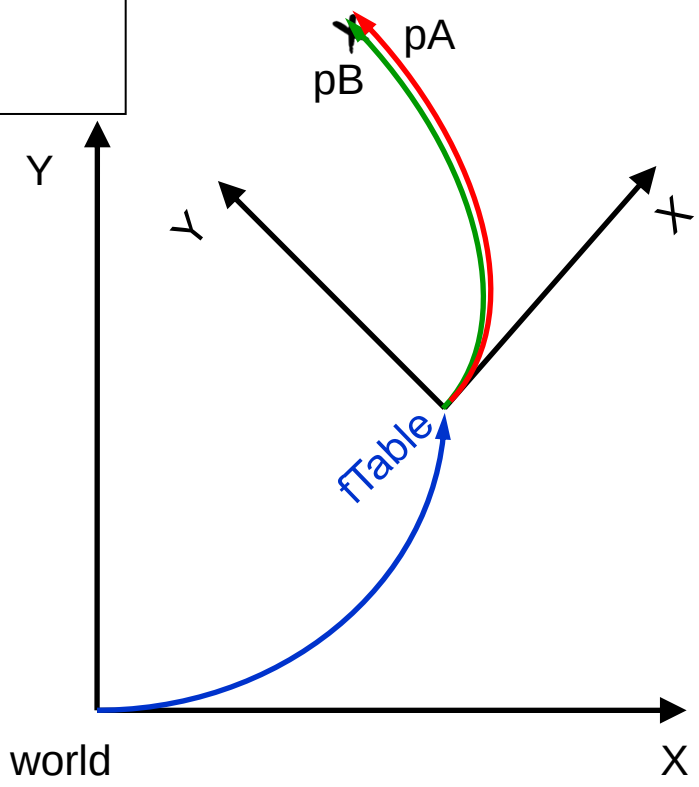


OPERACJE NA ZMIENNYCH POINT – C.D.

Zadeklarowano:
frame fTable
point pA, pB

program main()
begin
pA = pB
end

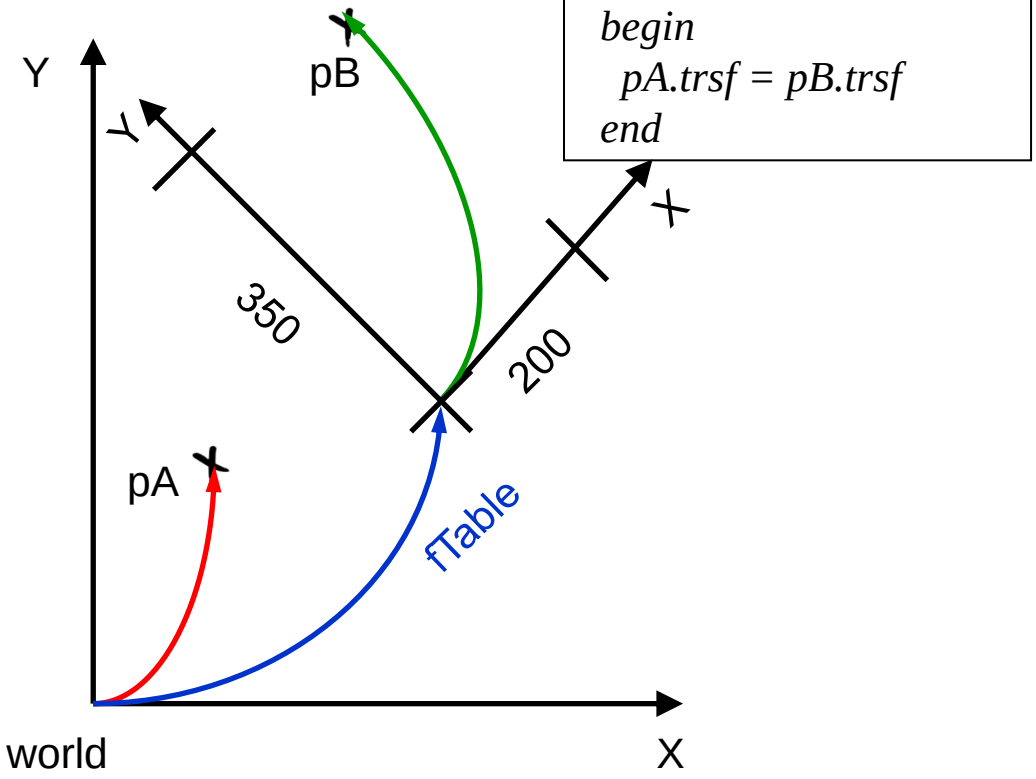




OPERACJE NA ZMIENNYCH POINT – C.D.

Zadeklarowano:
frame fTable
point pA, pB

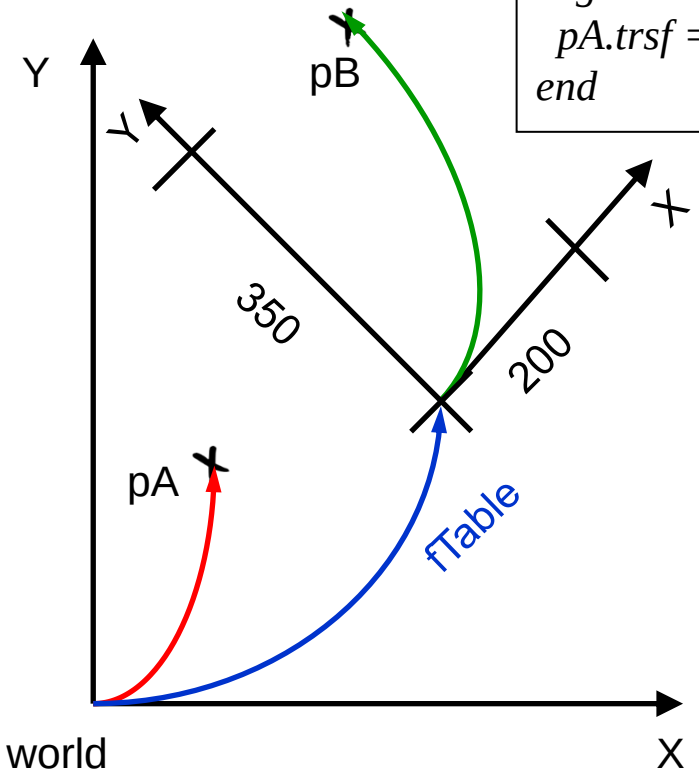
program main()
begin
pA.trsf = pB.trsf
end

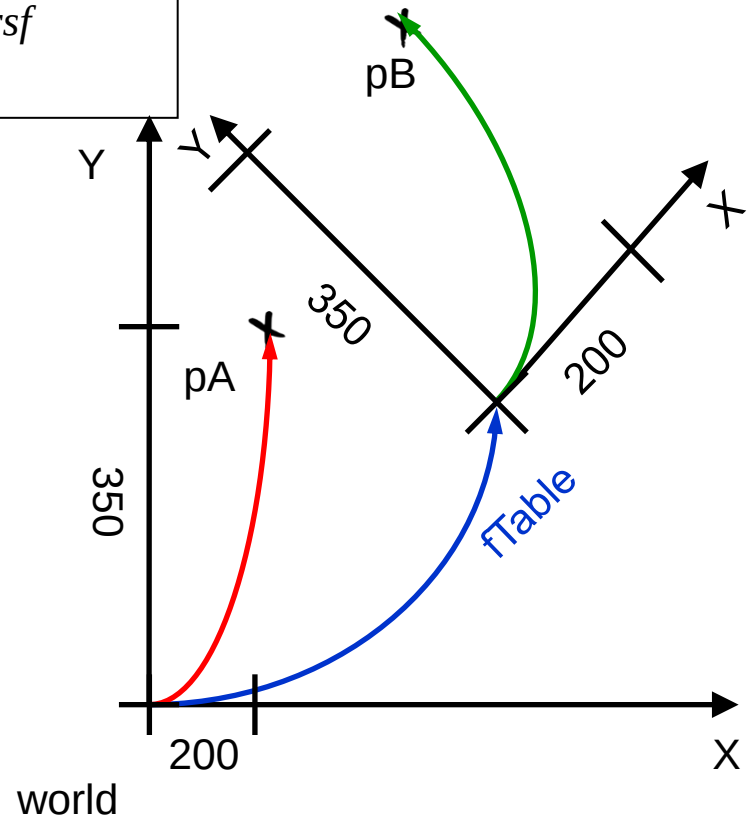


OPERACJE NA ZMIENNYCH POINT – C.D.

Zadeklarowano:
frame fTable
point pA, pB

program main()
begin
pA.trsf = pB.trsf
end

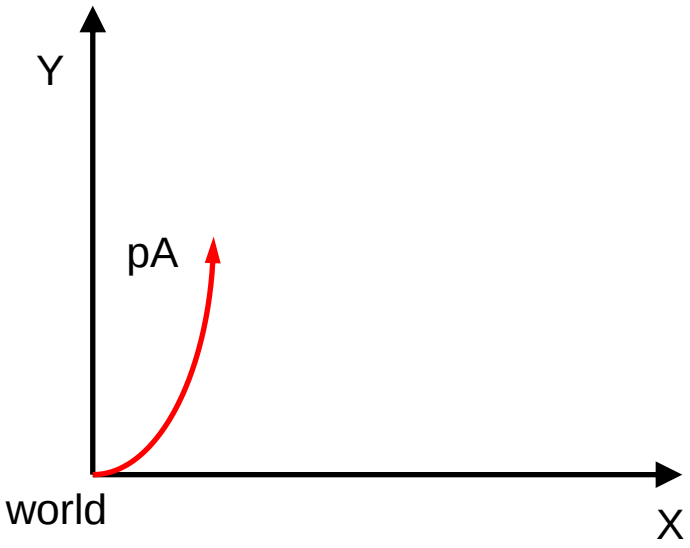




OPERACJE NA ZMIENNYCH POINT – C.D.

Zadeklarowano:
point pA

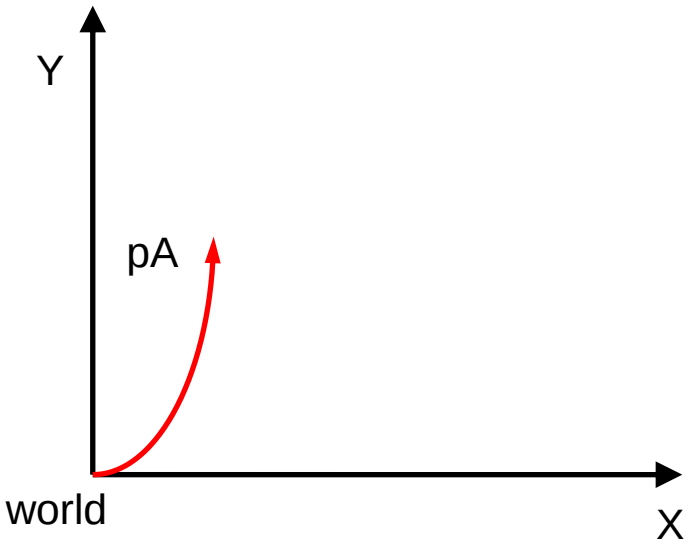
```
program main()
begin
  pA.trsf.x = pA.trsf.x + 100
end
```

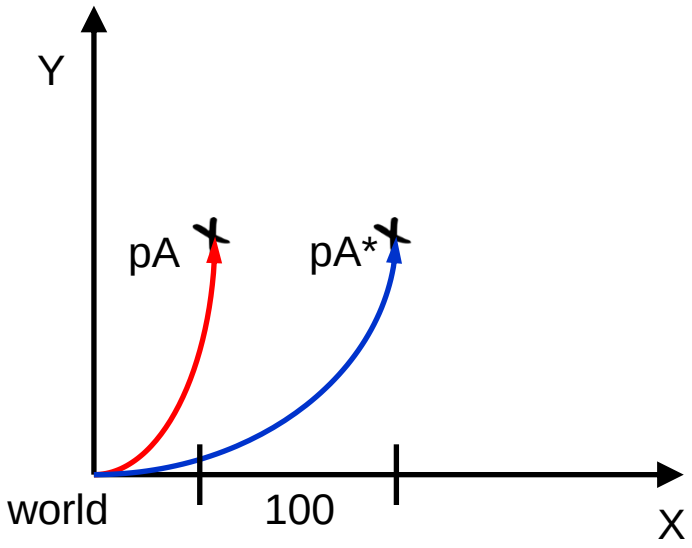


OPERACJE NA ZMIENNYCH POINT – C.D.

Zadeklarowano:
point pA

```
program main()
begin
  pA.trsf.x = pA.trsf.x + 100
end
```





PARAMETRY TOOL

TOOL .trsf .gripper .otime .ctime Połączone z narzędziem (niewodoczne) = != ==	TOOL grip : Możliwe takie same operacje jak dla punktów I transformat grip.gripper=true komendy połączone z wyjściem false grip.otime=0.2 grip.ctime=grip.otime Kopiowanie narzędzia : grip1=grip2

PARAMETRY JOINT/FRAME

FRAME .trsf = == !=	Takie same operacje jak na punktach
JOINT = == != > < - +	JOINT start start={0,10,20,30,40 ,50} lub start.j1=0 start.j2=10 start.j3=20 start.j4=start.j2+start.j3 start.j5=40 start.j6=50

PARAMETRY CONFIG

CONFIG .shoulder .elbow .wrist = == !=	CONFIG conf : conf.shoulder = righty / lefty / ssame / sfree conf.elbow=epositive jt3 > 0 enegative jt3 < 0 esame / efree conf.wrist=wpositive jt5>=0 wnegative jt5 < 0 wsame / wfree conf={righty,epositive,wnegative} POINT pta ptb : Zapisanie konfiguracji punktu : conf= pta.config Kopiowanie konfiguracji punktu : pta.config=ptb.config Wymuszenie konfiguracji w punkcie : pta.config.wrist=wpositive

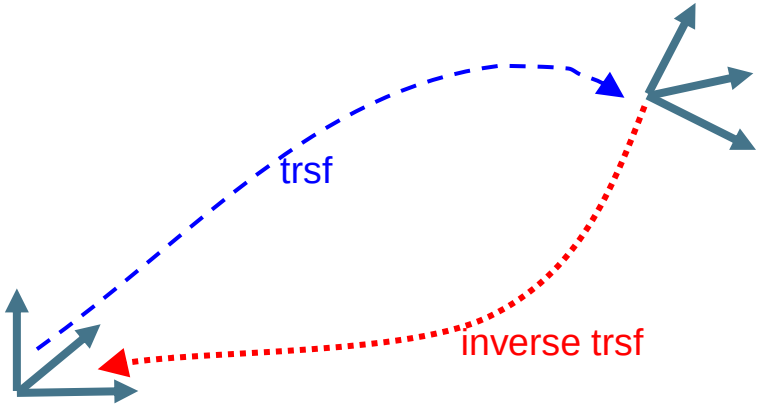
PARAMETRY TRSF

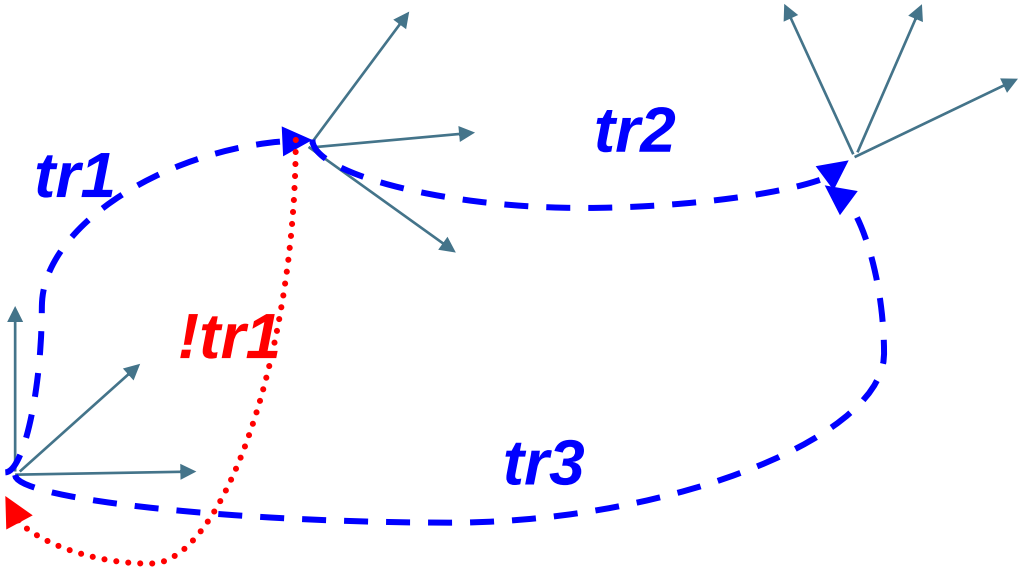
TRSF	TRSF t1 : TRSF t2 : TRSF t3 :
=	Złożenie transformat : *
==	t1=t2 * t3
!=	Odwrócenie transformat : !
	t1= ! t2

OPERACJE NA TRSF

Transformacja odwrotna

Strzałki w przeciwnym kierunku



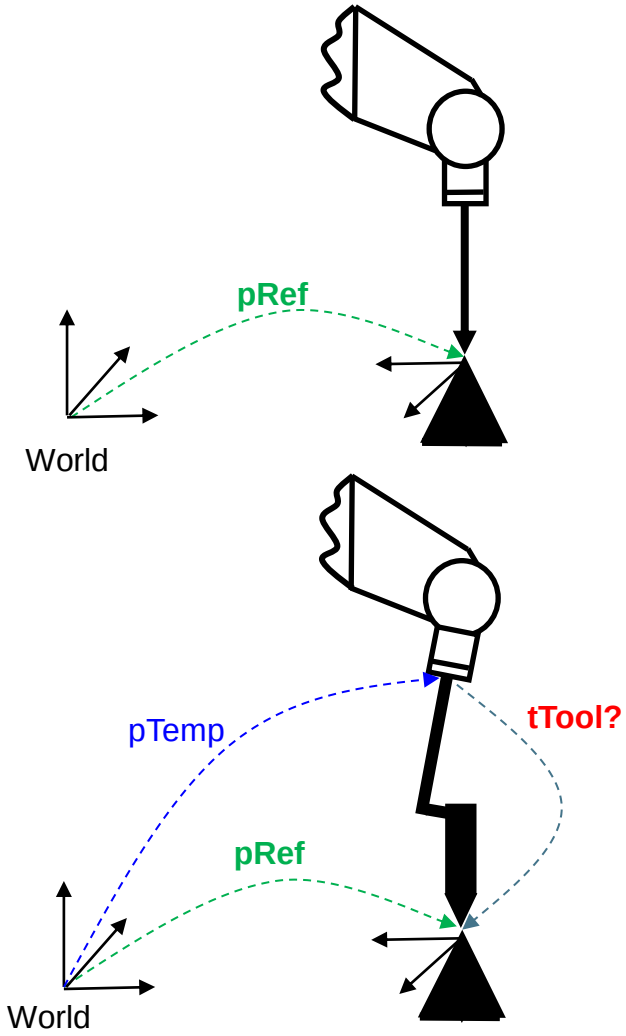


tr3=tr1*tr2

lub

tr2 =!tr1*tr3

Kalkulacja narzędzia przez program



1. Nauka punktu referencyjnego, znanym narzędziem:

   pRef=here(tPointer,world)

2. Nauka punktu tymczasowego niezdefiniowanym narzędziem:

   pTemp=here(flange,world)

3. Obliczenie układu narzędzia:

   tTool.trsf = !pTemp.trsf *pRef.trsf

- Uwaga: wartość **RZ** nie jest dobrze zdefiniowana.
- Rozwiązanie: zamiana **pRef** na układ współrzędnych **fRef** nauczony za pomocą 3 punktów.

OPERACJE NA JOINT

- Odczyt aktualnej pozycji Joint:

  jPosition=herej()

- Sprawdzenie czy pozycja jStart jest osiągalna:

  bOk = isInRange(jStart)

- Odczyt konfiguracji pozycji jPosition:

  conf=config(jPosition)

  - conf jest zmienną typu config np. pX.config=config(jX)

PRZYKŁAD JOINT: ROBOT W POZYCJI JSTART

```
program main()
begin
  if abs( herej() – jStart ) < {
2,2,2,10,20,20 }
    putln(“Robot in start position“)
  else
    putln(“Robot not in start position“)
  endIf
end
```

Opis:

- **herej()** zwraca aktualną pozycję robota w zmiennej joint (stopnie)
- Poprzez użycie **abs()** zadane zakresy przyjmują wartości **±**

Błąd!

```
program main()
begin
  if abs( herej() – jStart ) > {
2,2,2,10,20,20 }
    putln(“Robot not in start position“)
  else
    putln(“Robot in start position “)
  endIf
end
```

Obliczenie:

(actPos.j1 – jStart.j1) > 2 & (actPos.j2 – jStart.j2) > 2 & … & (actPos.j6 – jStart.j6) > 20)

→ Tylko gdy wszystkie osie są poza tolerancjami, kod działa poprawnie

OPERACJE NA POINT

- Odczyt aktualnej pozycji kartezjańskiej w układzie współrzędnych **fRef** za pomocą narzędzia **tGrip**:

```
pt = here(tGrip,fRef)
```

- Obliczenie odległości pomiędzy 2 punktami **pt1** i **pt2**:

```
nDist = distance(pt1, pt2)
```

- Obliczenie współrzędnych punktu **pt** w układzie współrzędnych **fRef**:

```
trCoord = position(pt, fRef)
```

Uwaga! Coord jest zmienną trsf

KONWERSJA JOINT <-> POINT

- Obliczenie punktu kartezjańskiego **pt** równego pozycji joint **jPosition** w układzie **fRef** z narzędziem **tTool**:

```
pt =jointToPoint(tTool, fRef, jPosition)
```

- Obliczenie punktu joint **jResult** równego punktowi kartezjańskiemu **pPosition** za pomocą narzędzia **tTool** i konfiguracji określonej w punkcie joint **jInitial**:

```
bOk= pointToJoint(tTool, jInitial, pPosition, jResult)
```

  - **bOk** = false jeżeli nie da się obliczyć pozycji

ROBOTICS

STÄUBLI

- CZĘŚĆ 3 – PROGRAMISTA CS9

ZMIENNE LOKALNE ORAZ PRZEKAZYWANIE PARAMETRÓW





ZMIENNE GLOBALNE / LOKALNE

Zmienne globalne:

Wspólne dla wszystkich programów

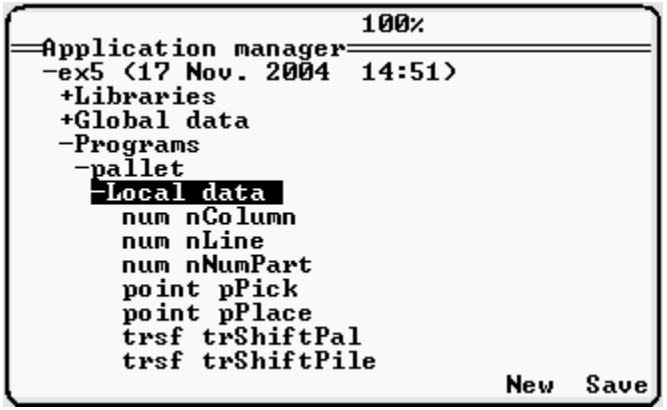
Ryzyko wystąpienia konfliktu!

Zmienne lokalne:

Niezależne od innych programów

konieczne przekazywanie do innych programów

ZMIENNE LOKALNE



Zmienne lokalne muszą być zdefiniowane w programie

Kopia istniejącej zmiennej

pLocal = pPick

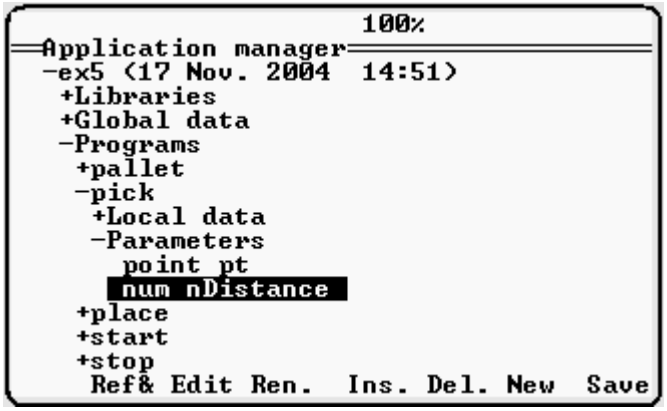
Wygenerowana przez obliczenia lub funkcję

nLocalValue = clock()

pLocal = here(tGrip, world)

PRZEKAZYWANIE PARAMETRÓW

```
program main()
nVal =10
call subprog(nVal)
```




```
program subprog(nDistance)
put(nDistance)
```

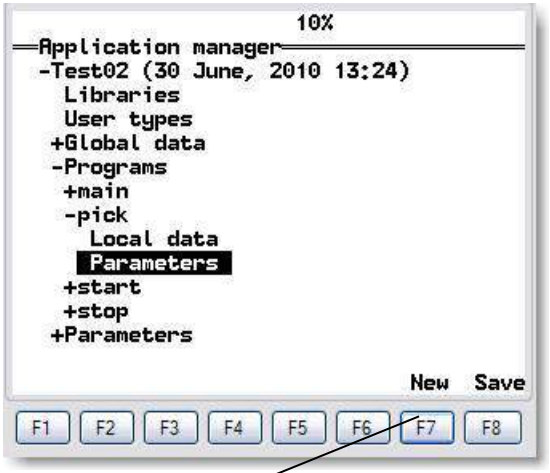
==> wyświetla 10

Tyle parametrów, ile jest potrzebne

! Uwaga ! Na liczbę i kolejność parametrów

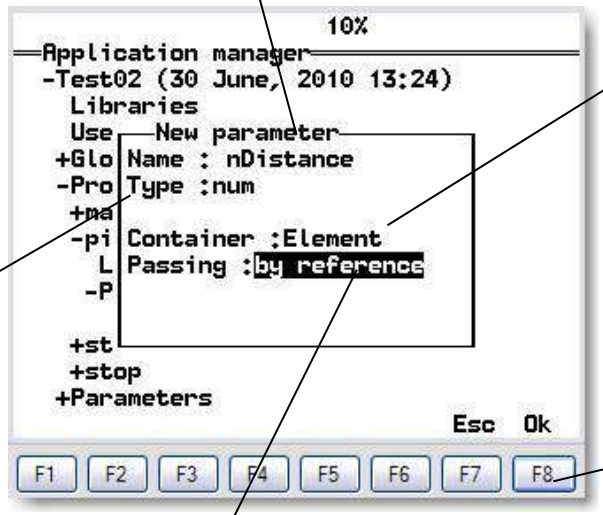
! Uwaga ! Parametry muszą być tego samego typu

TWORZENIE PARAMETRU



1 Tworzenie nowego parametru

2 Nazwa parametru



3 Typ

4 Rodzaj:
- Array
- Collection
- Element (Array o rozmiarze 1)

5 przekazywanie:
- przez wartość
- przez referencję

6 Ok: potwierdzenie

PRZEKAZYWANIE PARAMETRÓW „PRZEZ WARTOŚĆ”

Wywołanie podprogramu ***muliplication()***

- System tworzy zmienne parametrów **x_nValue**1 oraz **x_nValue2**
- Liczba **5** zostanie zapisana w parametrze **x_nValue1**
- Liczba **3** zostanie zapisana w parametrze **x_nValue2**

Jak tylko podprogram się wykona parametry **x_nValue1** and **x_nValue2** zostaną usunięte.

→ Wynik działania podprogramu: **7** w parametrze **x_nValue1** nie jest przekazywana do programu głównego!

```
Zadeklarowano:
num nNumber1
num nResult
num nCounter

program main()
begin
  nResult=0
  nNumber1 = 5
  call multiplication(nNumber1, 3)
  put(“nResult has the value “)
 putln(nResult)
end
```

```
program multiplication(num x_nValue1, num x_nValue2)
begin
 nResult = 0
 for nCounter = 1 to x_nValue1
   nResult = nResult + x_nValue2
 endFor
 x_nValue1 = 7
end
```

PRZEKAZYWANIE PARAMETRÓW „PRZEZ REFERENCJE”

Wywołanie podprogramu ***muliplication()***

- System tworzy zmienne parametrów **x_nValue1** oraz **x_nValue2**
- Liczba **5** zostanie zapisana w parametrze **x_nValue1**
- Liczba **3** zostanie zapisana w parametrze **x_nValue2**
- Zmienna **x_nResult** wskazuje na to samą zmienną co **nResul**t

  → Jeżeli **x_nResult** ulega zmianie, **nResult** także jest zmieniana

Jak tylko program *multiplication()* się zakończy, parametry **x_nValue1**, **x_nValue2** i wskaźnik **x_nResult** będą usunięte.

```
Zadeklarowano:
num nValue1
num nResult
num nCounter

program main()
begin
  nValue1 = 5
  call multiplication(nValue1, 3, nResult)
  put(“nResult has the value “)
  putln(nResult)
end
```

```
program multiplication(num x_nValue1, num x_nValue2, num& x_nResult)
begin
  x_nResult = 0
  for nCounter = 1 to x_nValue1
    x_nResult = x_nResult + x_nValue2
  endFor
end
```

PRZEKAZYWANIE PARAMETRÓW – PODSUMOWANIE

STÄUBLI

- **przez wartość:** num x_nWert1
  - Tworzy lokalną kopię zmiennej
  - Zmiana wartości zmiennej w podprogramie nie ma wpływu na wartość w programie głównym

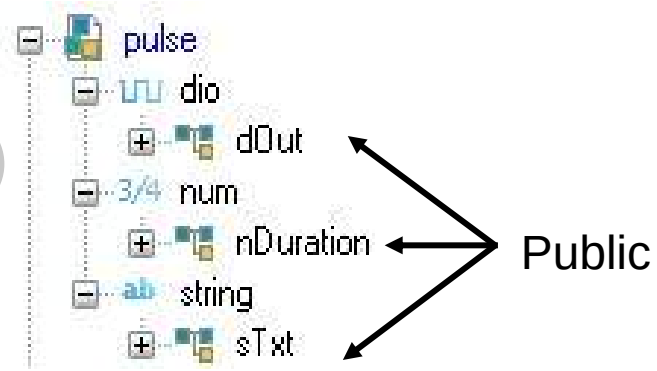
W programie main():	nNumber1 = 5
W programie multiplication():	x_nValue1 = 7
"przekazana" wartość w main():	nResult = 15

- **przez referencję:** num**&** x_nResult
  - Brak kopii lokalnej, jedna zmienna pod dwoma nazwami
  - Zmiana wartości zmiennej w podprogramie zmienia jej wartość w programie głównym

W programie main():	nResult = ?
W programie multiplication():	x_nResult = 15
"przekazana" wartość w main():	nResult = 15

ZMIENNE DEFINIOWANE PRZEZ UŻYTKOWNIKA

1

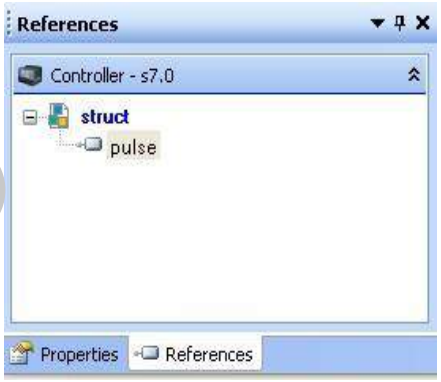


1. Typ zmiennych definiowany podobnie do bibliotek

➔ 1 aplikacja ze zmiennymi publicznymi: (np. *pulse*)

2. Typ jest deklarowany w aplikacji, która będzie z niego korzystać (np. **struct**)
3. Zmienne tego typu są definiowane w aplikacji (np. *Flash*)
4. Składnia typu: menu zmiennych

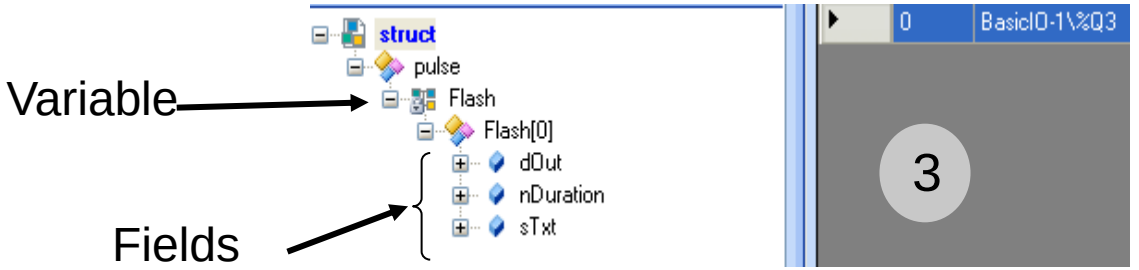
2



Deklaracja typu w menu Referencji

4

```
Flash.dOut=true
delay(Flash.nDuration)
Flash.dOut=false
logMsg(Flash.sTxt)
```

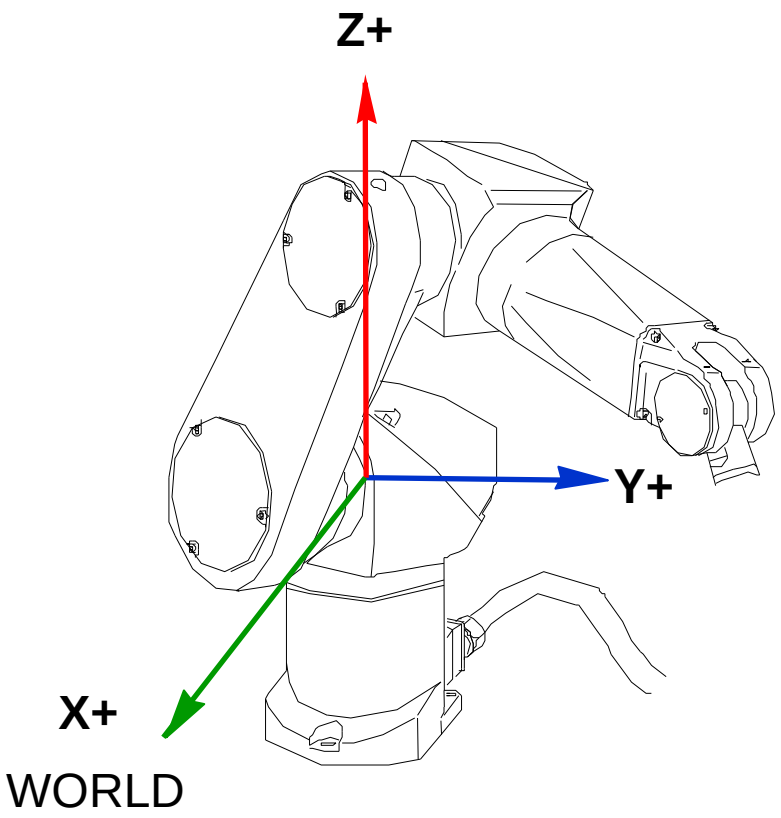


CZĘŚĆ 3 – PROGRAMISTA CS9

UKŁADY WSPÓŁRZĘDNYCH I PALETYZACJA – C.D.

WPROWADZENIE STRONY 163-171

ZMIENNA TYPU FRAME



Układ współrzędnych „world“

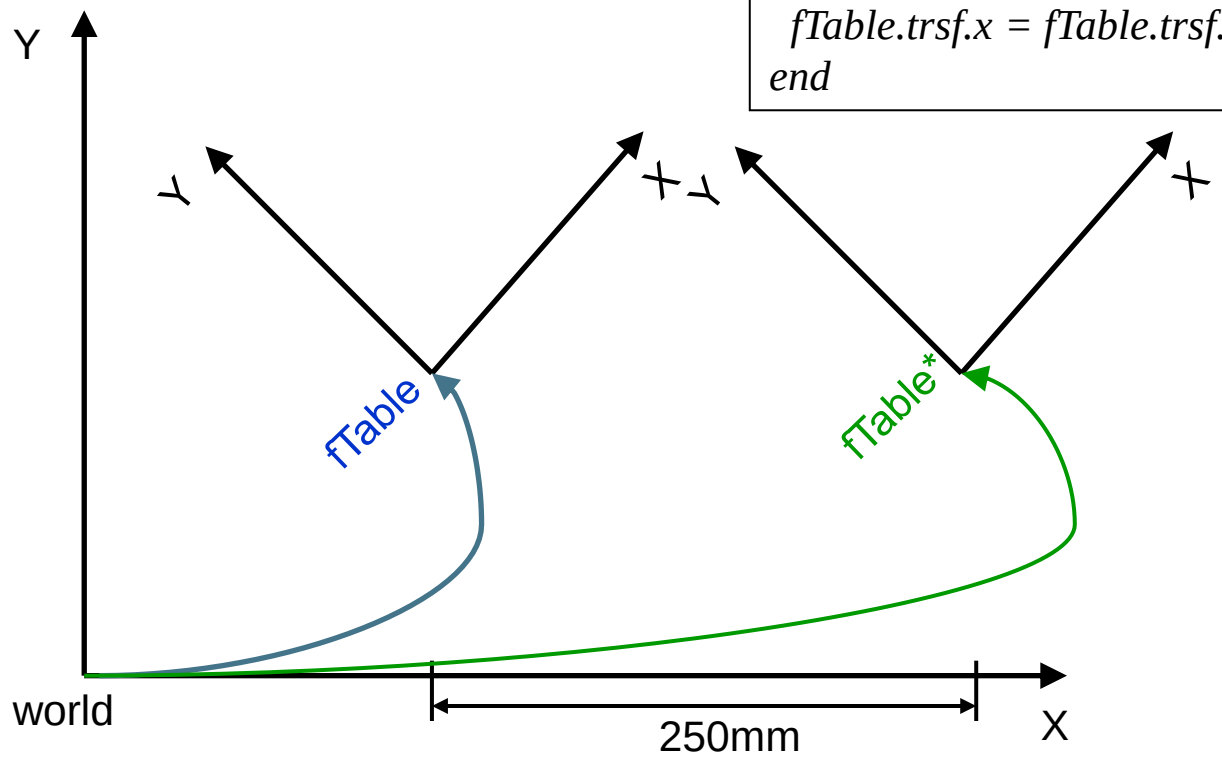
- Kartezjański układ współrzędnych w podstawie robota
- Nie może być zmieniany

Wszystkie pozostałe układy współrzędnych są z nim powiązane

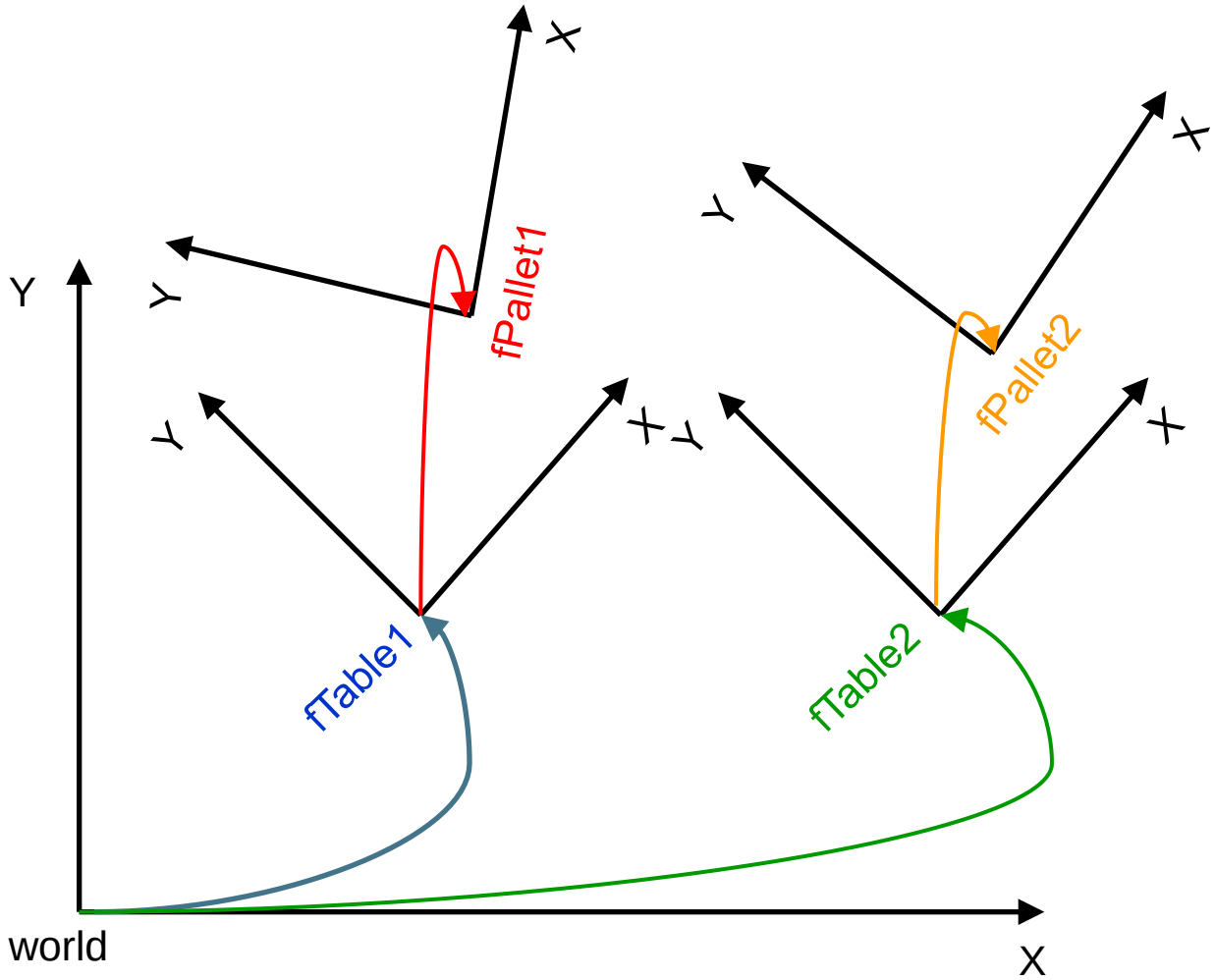
OPERACJE NA ZMIENNEJ FRAME

Zadeklarowano:
frame fTable

program main()
begin
fTable.trsf.x = fTable.trsf.x + 250
end



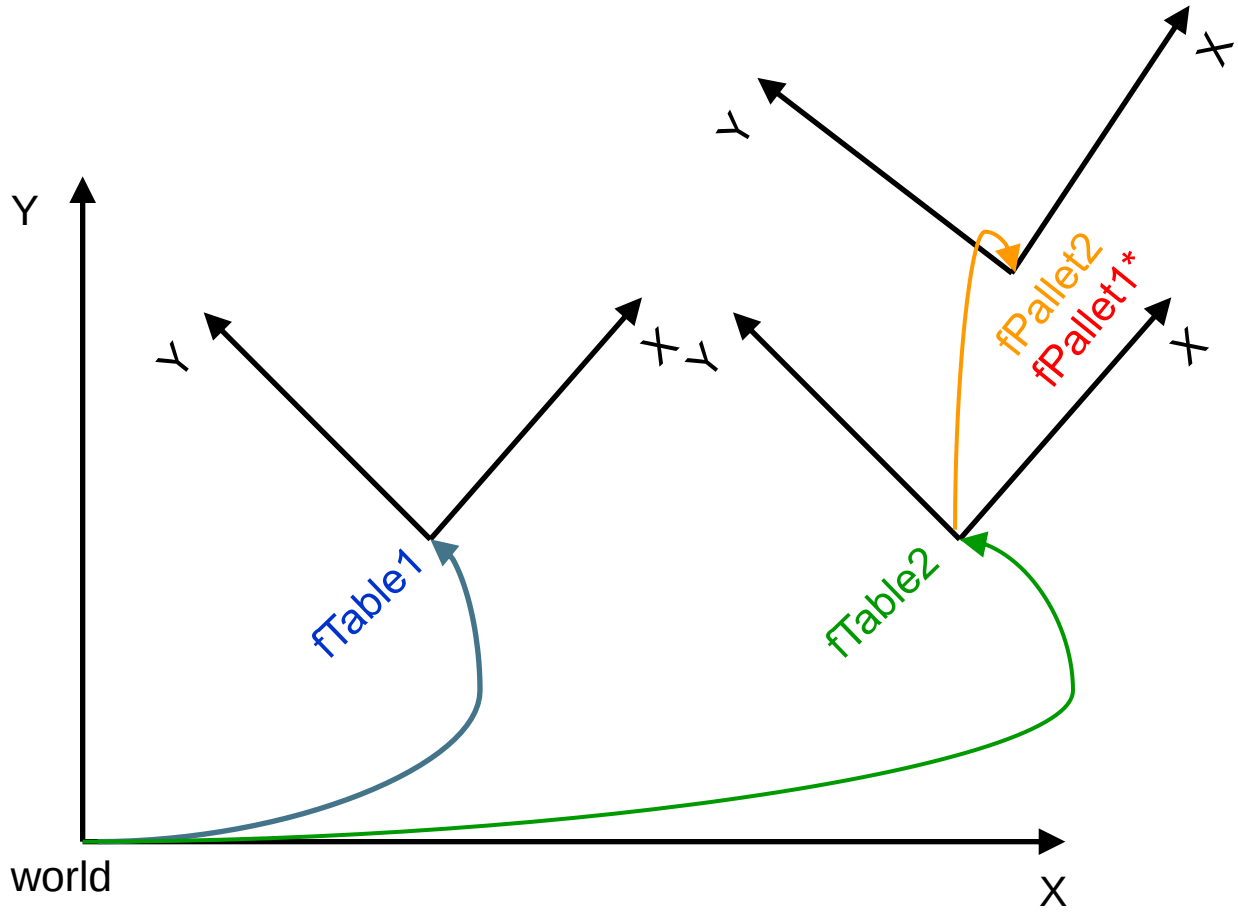
OPERACJE NA ZMIENNEJ FRAME



Zadeklarowano:
frame fTable1
frame fTable2
frame fPallet1
frame fPallet2

program main()
begin
fPallet1 = fPallet2
end

OPERACJE NA ZMIENNEJ FRAME



Zadeklarowano:
frame fTable1
frame fTable2
frame fPallet1
frame fPallet2

```
program main()
begin
  fPallet1 = fPallet2
end
```

Układy współrzędnych i paletyzacja – ćwiczenie

Ćwiczenie nr 10: Paletyzacja: punkt <-> paleta – zadanie nr 4

Robot pobiera elementy ze stołu i układa je na palecie.

Ćwiczenie nr 11: Paletyzacja: paleta <-> punkt – zadanie nr 5

Robot pobiera elementy z palety i układa je na stole.

Ćwiczenie nr 12: Paletyzacja: paleta <-> paleta – zadanie nr 6

Robot pobiera elementy z pierwszej palety i układa je na drugiej palecie.

1. Proszę nauczyć układ współrzędnych wykorzystując odpowiednie punkty.
2. Proszę wykorzystać odpowiednie narzędzia (do nauki układu współrzędnych oraz chwytak)
3. Jedna z palet ustawiona pod kątem do układu współrzędnych robota druga płaska – dla robota 6-cio osiowego. Dla robota 4 osiowego użyć dwóch paletek płaskich.
4. Odstęp między detalami wynosi 50 mm wysokość detalu wynosi 40 mm.
5. Zaleca się wykorzystanie podprogramów (funkcji), zmiennych oraz pętli.

ROBOTICS

STÄUBLI

- CZĘŚĆ 3 – PROGRAMISTA CS9

POLECENIA SYSTEMOWE I WIELOZADANIOWOŚĆ





TIMER: CLOCK()

nVal = clock()

Zwraca aktualną wartość zegara systemowego w sekundach

Precyzja = ms

Wartość 0 przyjmuje podczas włączenia kontrolera

Przewinięcie po około 49 dniach

Pomiar czasu cyklu:

```
nStart=clock()
……..
nTime=clock()-nStart
```

getDate(): Zwraca datę oraz godzinę ustawioną w systemie

INSTRUKCJE SYSTEMOWE

isPowered(): zwraca TRUE jeżeli są załączone serwonapędy;

isCalibrated(): zwracaTRUE jeżeli robot jest skalibrowany (jeżeli robot nie jest skalibrowany, wyświetlany jest komunikat po uruchomieniu kontrolera);

isSettled(): zwraca TRUE jeżeli robot zatrzyma się po wykonaniu ruchu lub po wyłączeniu napędów lub po naciśnięciu przycisku MOVE/HOLD;

isEmpty(): zwraca TRUE jeżeli stos poleceń ruchu jest pusty = wszystkie ruchy zostały wykonane;

esStatus(): zwraca stan układu bezpieczeństwa = 2 jeżeli E-STOP jest aktywny;

INSTRUKCJE SYSTEMOWE – C.D.

workingMode(): zwraca aktualny tryb pracy:

1: tryb ręczny

2: tryb testowy

3: tryb automatyczny lokalny

4: tryb automatyczny zdalny

0: zmiana trybu pracy

disablePower(): wyłącza serwonapędy;

enablePower(): załącza serwonapędy (tylko w trybie automatycznym zdalnym)

getMonitorSpeed(): zwraca aktualnie ustawioną prędkość robota w %;

setMonitorSpeed(num): zmienia prędkość w trybie automatycznym zdalnym;

POLECENIE WORKINGMODE()

```
program main()
begin
  // sprawdza tryb pracy
  nWorkingMode=workingMode()

  // sprawdza tryb pracy
  // i uzyskuje dodatkowe informacje
(state)

nWorkingMode=workingMode(nMoreI
nfo)
end
```

Mode	State	workingmode	Explanation
0	0	Invalid workingmode	-
1	0	manual	Programmed move
	1		Connection move
	2		Joint
	3		Translation move (Frame)
	4		Translation move (Tool)
	5		JogInterface (Point)
	6		Hold
2	0	100% Test	Programmed move (<250mm/s)
	1		Connection move (<250mm/s)
	2		Programmed move (>250mm/s)
	3		Hold
3	0	Local automatic	Move (programmed move)
	1		Move (connection move)
	2		Hold
4	0	Remote automatic	Move (programmed move)
	1		Move (connection move)
	2		Hold

PROFILE UŻYTKOWNIKÓW – DOSTĘP Z POZIOMU VAL3

STÄUBLI

string getProfile()

- zwraca zmienną string z nazwą aktualnie wybranego użytkownika

num setProfile(string sProfile, string sPassword)

- zmienia na wybrany profil użytkownika
- zwracane wartości

0: wybrano profil użytkownika

-1: profil nie istnieje

-2: wprowadzono złe hasło

-3: nie można użyć profilu „Stäubli”

-4: aktualny profil to „Stäubli” i nie można go zmienić

E-STOP

Sygnały E-stop :

- Naciśnięto Estop na MCP lub WMS
- Naciśnięto E-Stop w celi
- Przełącznik zwalniania hamulców w pozycji innej niż 0
- Załączona krańcówka ogranicznika ruchu

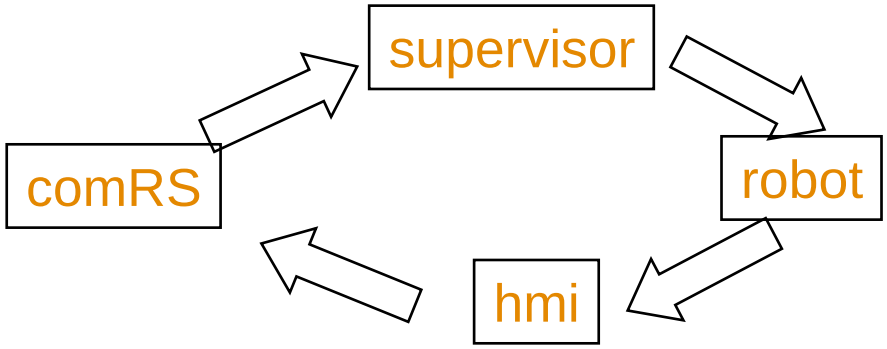
esStatus() zwraca status układu bezpieczeństwa

0	Nie przerwano łańcucha bezpieczeństwa
1	Zwolniono E-stop, lecz nie załączono zasilania ramienia
2	Przerwano łańcuch bezpieczeństwa.

W trybie ręcznym po wciśnięciu przycisku E-stop, MCP musi być umieszczony na podstawie. W przeciwnym wypadku zasilanie ramienia nie zostanie włączone. Podstawa MCP musi być umieszczona poza celą.

WIELOZADANIOWOŚĆ

- **Wielozadaniowość** = kilka programów może działać w tym samym czasie.
- Pozwala na wykonywanie kilku funkcji za pomocą niezależnych programów.
- Współdzielony czas CPU, ale szybkie przełączanie pomiędzy zadaniami.
- Komunikacja i synchronizacja pomiędzy programami odbywa się za pomocą zmiennych globalnych i sygnałów I/O.

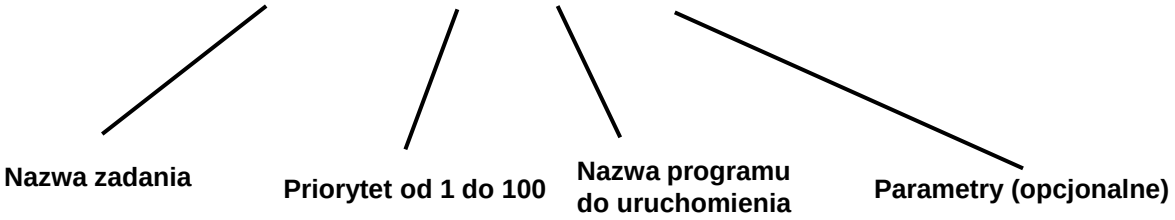


- Przykład :
  - Podprogram ruchów ramienia: «robot»
  - Podprogram nadzoru (tryb pracy celi , zarządzanie błędami, inicjalizowanie zmiennych: «supervisor »
  - Podprogram ekranów użytkownika: «hmi»
  - Komunikacja z urządzeniami peryferyjnymi: RS232, Eth socket: «comRS»

WIELOZADANIOWOŚĆ C.D.

- Brak limitów ilości jednocześnie wykonywanych zadań (ograniczeniem jest pamięć operacyjna)
- Każde zadanie ma swoją nazwę i priorytet
- Priorytet określa maksymalną ilość linii do wykonania, potem następuję przejście do następnego zadania.

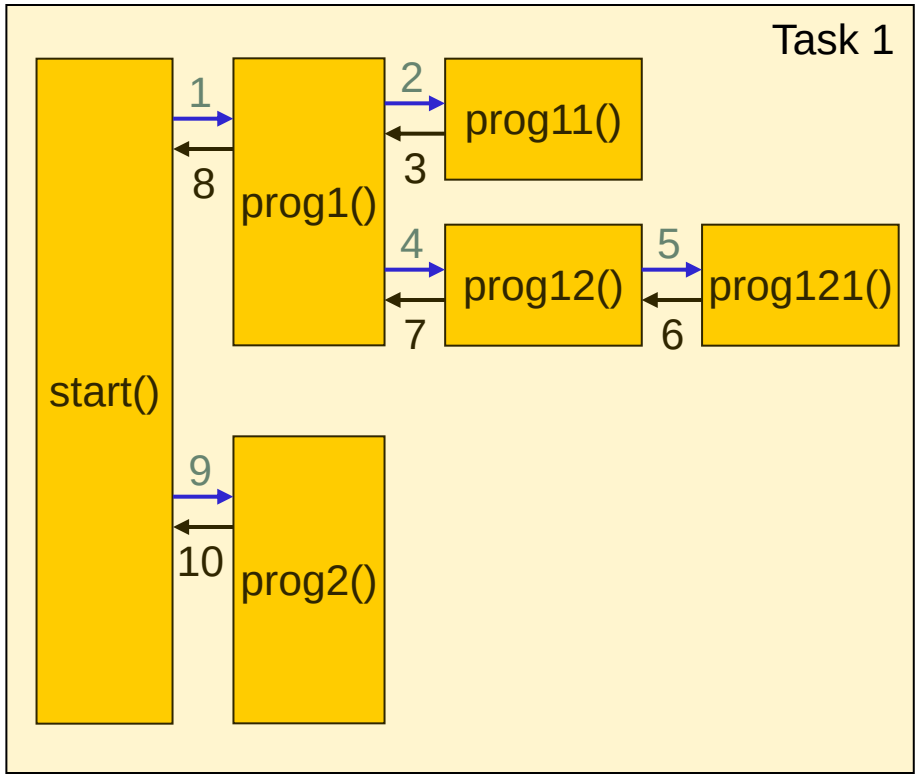
taskCreate “security”,100,prog(params) tworzy nowego taska



- Aplikacja jest uruchomiona tak długo jak jej zadania są uruchomione.

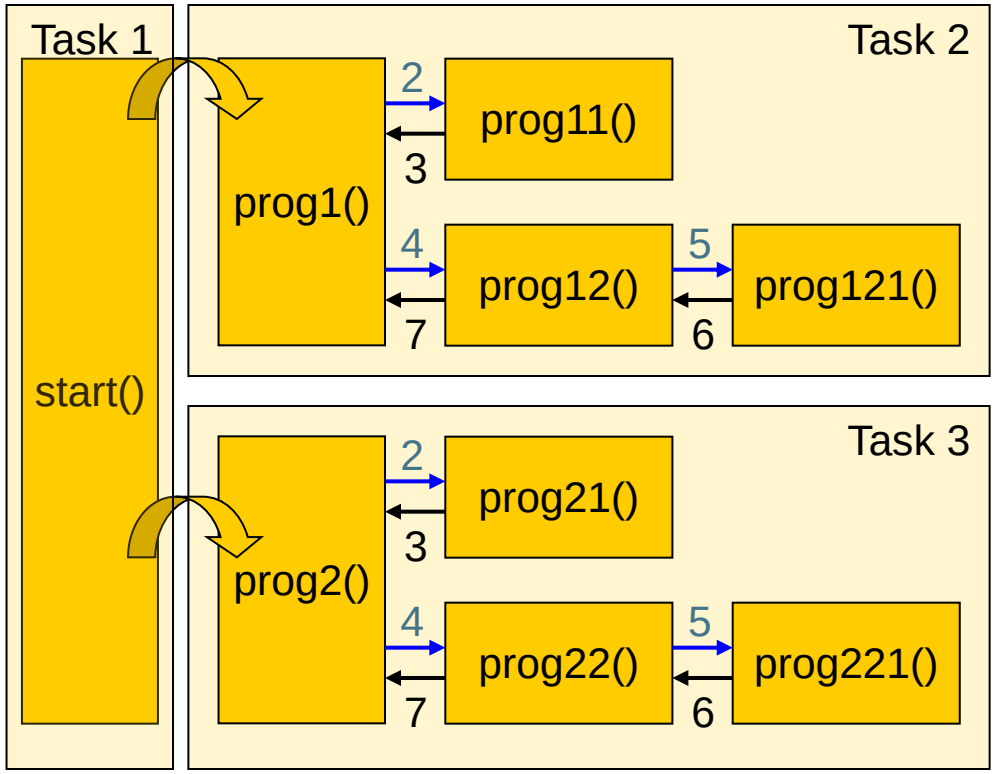
PORÓWNANIE CALL() I TASKCREATE()

Przebieg programów **call()**:



Wykonanie sekwencyjne,
Programy wykonywane jeden po drugim
Jeden potok wykonywania programów

Przebieg programów **taskCreate()** & **call()**:



Wykonywanie równoległe,
Wiele potoków wykonywania programów

PRIORYTETY W VAL3

STÄUBLI

taskCreate “secu”, 10, prog(Parameter)

- Priorytet określa, jak wiele linii programu może być wykonanych za jednym razem, zanim inne zadanie zostanie wykonane.
- Priorytet może przyjąć wartości od 1 do 100
- Poniższe instrukcje powodują przejście do następnego zadania:
  - watch(), delay(), wait(), waitEndMove(), setMutex()
  - open(), close()
  - get()
  - disablePower()
  - taskResume(), taskKill()
  - libLoad(), libSave(), libDelete(), libList(), setProfile()
  - Read/Write to SIO: sioGet(), sioSet() i operator =

WYKONYWANIE ZADAŃ ASYNCHRONICZNYCH

Zadania tworzone przez **taskCreate()** są wykonywane niezależnie od czasu cyklu

Podczas jednego cyklu VAL3-cycle są wykonywane:

1. Odczyt wejść
2. Wykonanie wszystkich zadań VAL3-Tasks w zależności od priorytetu
3. Ustawienie wyjść

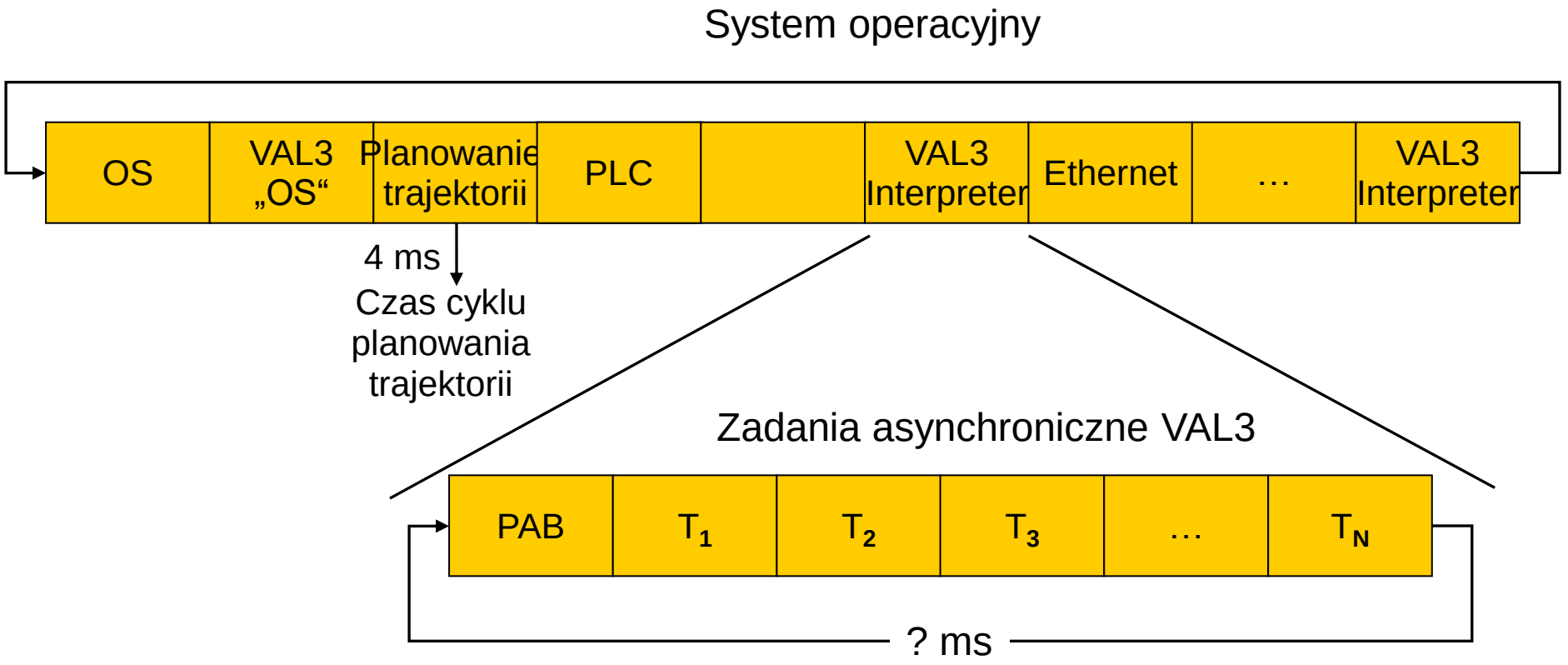
Na czas cyklu VAL3 mają wpływ:

- Ilość wykonywanych zadań
  → Im więcej tym dłuższy czas cyklu
- Priorytet zadań
  → Im większy tym dłuższy czas cyklu
- Instrukcje wait(), waitEndMove(), itp.
  → jeżeli nie ma oczekiwania, czas cyklu się skraca
- Ilość wejść/wyjść
  → Im więcej tym dłuższy czas cyklu

→ Czas cyklu nie jest stały i określony
→ Nieprzydatne do zadań cyklicznych np. tracking
→ Wykonuje się najszybciej jak to możliwe

VAL3 TASKS – ZADANIA VAL3

Zasada działania:



ZARZĄDZANIE ZADANIAMI

taskKill(“secu”) zakańcza zadanie

taskSuspend(“secu”) wstrzymuje wykonywanie zadania

taskResume(“secu”,jump) wznawia wykonywanie zadania

jeżeli jump =0 w obecnej linii

jeżeli jump >0 o X linii po

jeżeli jump <0 o X linii przed

taskStatus(“secu”) zwraca status zadania

(1:wykonywane, 0:zatrzymane,-1: nie istnieje)

inne wartości to pozostałe kody błędów

ZARZĄDZANIE ZADANIAMI – C.D.

taskStatus(“secu”)

Zwraca status zadania: “secu”

Code	Description
-1	There is no task created by this application or library with the specified name
0	No runtime error
1	The specified task is running
10	Invalid numerical calculation (division by zero).
11	Invalid numerical calculation (e.g.ln(-1))
20	Access to an array with an index that is larger than the array size.
21	Access to an array with a negative index.
29	Invalid task name. See taskCreate() instruction.
30	The specified name does not correspond to any VAL3 task.
31	A task with the same name already exists. See taskCreate instruction.
32	Only 2 different periods for synchronous tasks are supported. Change scheduling period.
40	Not enough memory space available.
41	Not enough memory space to run the task. See the run memory size.
60	Maximum instruction run time exceeded.
61	Internal VAL3 interpreter error
70	Invalid instruction parameter. See the corresponding instruction.
80	Uses data or a program from a library not loaded in the memory.
81	Incompatible kinematic: Use of a point/joint/config that is not compatible with the arm kinematic.
82	The reference frame or tool of a variable belongs to a library and is not accessible from the variable's scope (library not declared in the variable's project, or reference variable is private).
90	The task cannot resume from the location specified. See **taskResume()** instruction.
100	The speed specified in the motion descriptor is invalid (negative or too great).
101	The acceleration specified in the motion descriptor is invalid (negative or too great).
102	The deceleration specified in the motion descriptor is invalid (negative or too great).
103	The translation velocity specified in the motion descriptor is invalid (negative or too great).
104	The rotation velocity specified in the motion descriptor is invalid (negative or too great).
105	The reach parameter specified in the movement descriptor is invalid (negative).
106	The leave parameter specified in the movement descriptor is invalid (negative).
122	Attempt to write in a system input.
123	Use of a dio, aio or sio input/output not connected to a system input/output.
124	Attempt to access a protected system input/output
125	Read or write error on a dio, aio or sio (field bus error)
150	Cannot run this movement instruction: a previous movement request could not be completed (point out of reach, singularity, configuration problem, etc.)
153	Movement command not supported
154	Invalid movement instruction: check the movement descriptor.
160	Invalid flange tool coordinates
161	Invalid world tool coordinates
162	Use of a point without a reference frame. See Definition.
163	Use of a frame without a reference frame. See Definition.
164	Use of a tool without reference tool. See Definition.
165	Invalid frame or reference tool (global variable linked to a local variable)
250	No runtime licence for this instruction, or demo licence is over.

URUCHAMIANIE I ZATRZYMYWANIE PROGRAMÓW

Po naciśnięciu przycisku **RUN** wykonywane są instrukcje w programie **START**.

Po naciśnięciu przycisku **STOP** lub po zakończeniu aplikacji wykonywane są instrukcje w programie **STOP**.

Po naciśnięciu przycisku STOP zadanie START jest automatycznie zatrzymywane i niszczone, po wykonaniu programu STOP wszystkie zadania są niszczone.

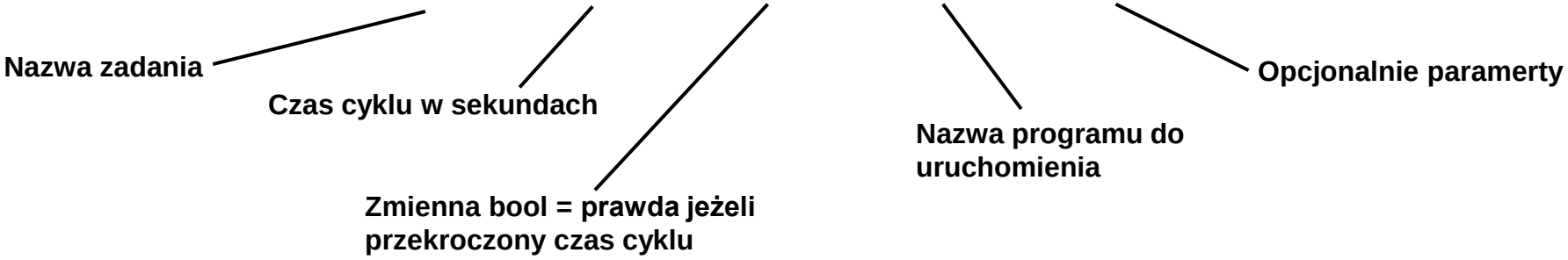
ZADANIA SYNCHRONICZNE

taskCreate tworzy zadanie asynchroniczne, wykonywane najszybciej jak to możliwe.

W niektórych przypadkach zachodzi konieczność wykonania zadania w określonym czasie: zbieranie danych, obliczenia powiązane z czasem rzeczywistym, śledzenie transporterów.

Zadania synchroniczne zarządzają czasem tak jak PLC => przerywają wykonywanie zadań asynchronicznych co określony czas

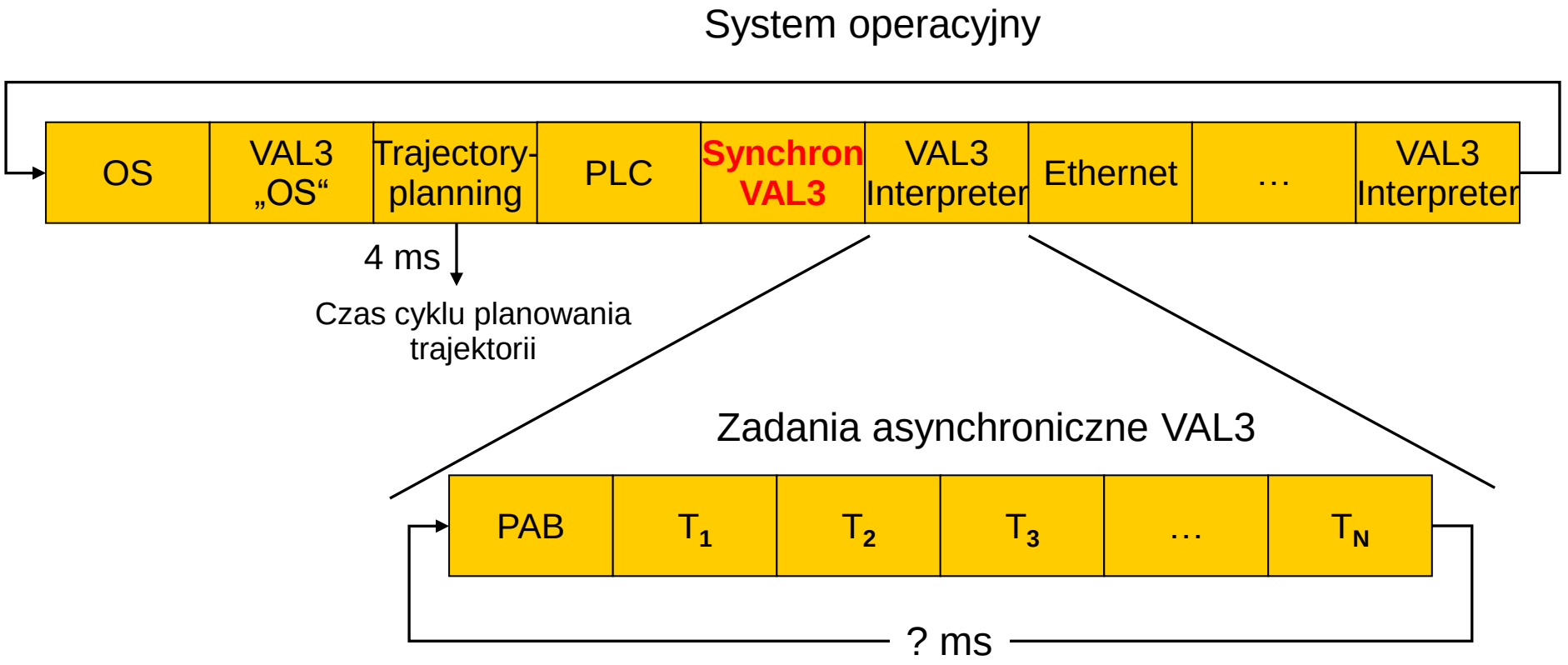
taskCreateSync “conveyor”, 0.004, bOverRun, prog(Parameter)



- Czas cyklu: wielokrotność 4 ms,
- Zezwala na przejście do innego zadania przez delay(0) (lub 3000 linii)
- Analiza czasu CPU w task manager w menu « info »

SYNCHRONICZNE I ASYNCHRONICZNE ZADANIA VAL3

Zasada działania:



ROBOTICS

STÄUBLI

- CZĘŚĆ 3 – PROGRAMISTA CS9

BIBLIOTEKI





CEL

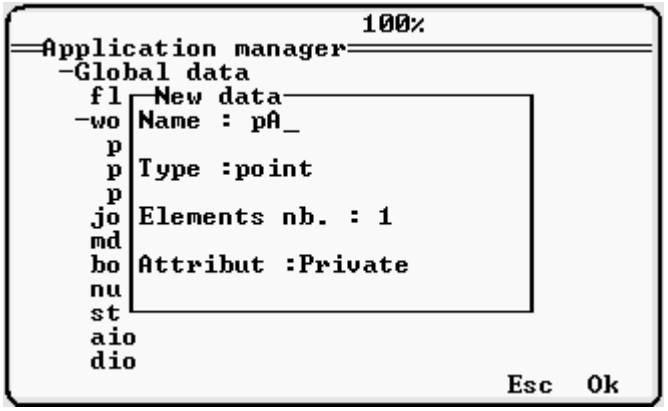
Wykorzystanie tych samych danych w wielu programach

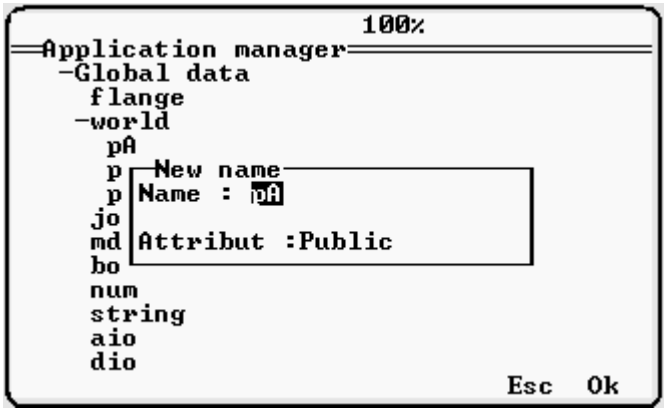
Przykład: biblioteki z programami wykorzystywanymi wielokrotnie w różnych podprogramach

Biblioteki z danymi do pojedynczej aplikacji wykorzystującej wiele referencji
1 unikalna aplikacja + 1 biblioteka z referencjami części

Biblioteka jest uproszczoną aplikacją, ALE trzeba zadeklarować zmienne/podprogramy, które będą wykorzystywane i widoczne dla innych aplikacji

EKSPORT DANYCH / PROGRAMÓW



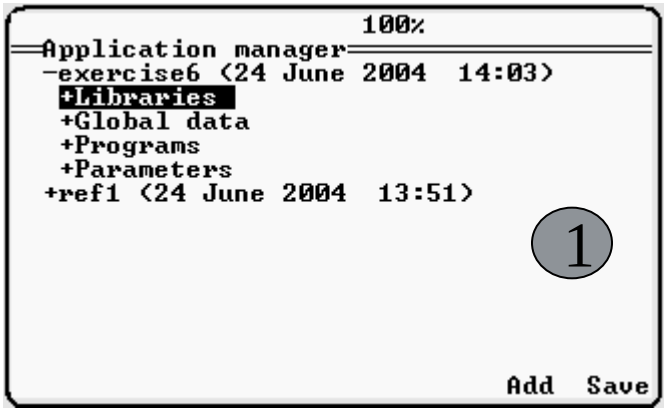


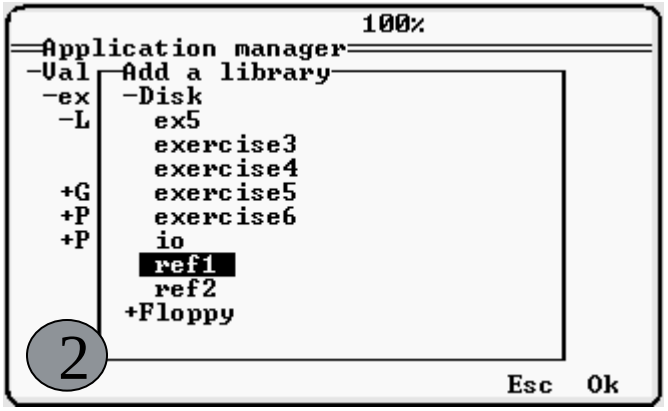
Zmiana atrybutu na **PUBLICZNY**

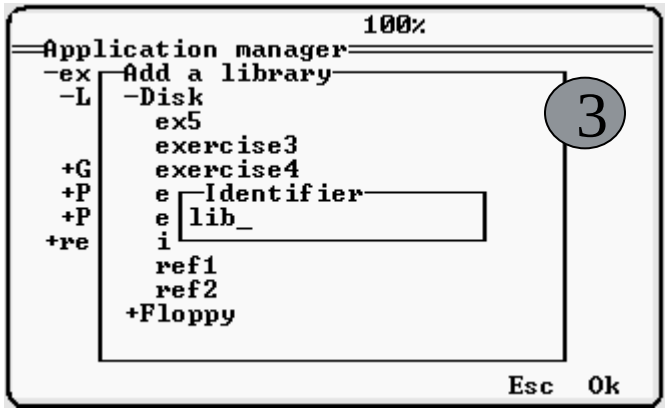
DEKLARACJA BIBLIOTEK

Interfejs biblioteki stanowią dane i programy w niej zawarte.

Interfejs biblioteki wykorzystywany w aplikacji musi zostać zadeklarowany.







Wybór biblioteki
Np: Ref1

Nazwa identyfikatora w aplikacji wykorzystującej bibliotekę np. libRef1

WYKORZYSTANIE BIBLIOTEK

• Wykorzystanie punktu **pA** zdefiniowanego jako **publiczny** w bibliotece Ref1

```
movej(libRef1:pA, tGripper, mFast)
```

• Wykorzystanie zmiennej numerycznej **X** zdefiniowanej jako **publiczna** w bibliotece Ref1

```
libRef1:nX=10
```

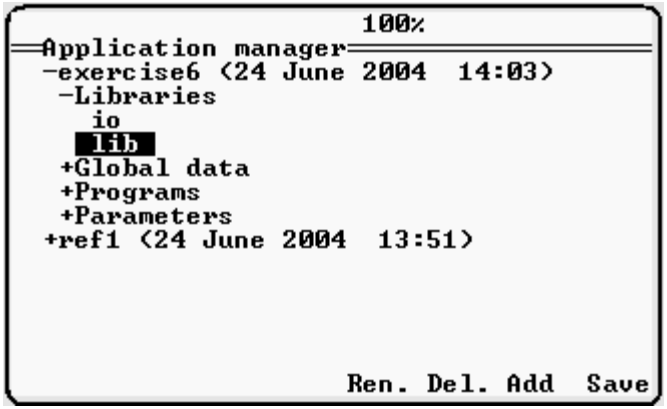
• Wywołanie programu **INIT** zdefiniowanego jako **publiczny** w bibliotece Ref1

```
call libRef1:init()
```

ŁADOWANIE BIBLIOTEK

Przykład aplikacji wykorzystującej kilka bibliotek z punktami **pA** w **ref1** oraz **ref2**:

```
nErr=lib:libload("ref1")          załadowanie biblioteki ref1
movej(lib:pA,tGrip,mFast)         wykorzystuje pA zapisanego w ref1
nErr=lib:libload("ref2")          załadowanie biblioteki ref2
movej(lib:pA,tGrip,mFast)         wykorzystanie pA zapisanego w ref2
```



Jedna nazwa zmiennej wspólna dla kilku bibliotek, zawierająca ten sam typ danych o innych wartościach.

ZAPISYWANIE DO BIBLIOTEK

Możliwość zapisu do bibliotek

nErr=lib:libSave()	zapisuje bibliotekę pod nazwą « **lib** » na dysku Flash
nErr=lib:libload("ref2")	załadowuje bibliotekę **ref2**
movej(lib:pA,flange,nom_speed)	wykorzystanie **pA** z biblioteki **ref2**
lib:nX=10	zmiana wartości zmiennej **nX** w bibliotece **ref2**
nErr=lib:libSave()	zmienna **nX** jest zapisana na dysku
nErr=lib:libload("ref1")	załadowanie biblioteki **ref1**
put(lib:nX)	wyświetlenie wartości zmiennej **nX** z biblioteki **ref1**
nErr=lib:libSave("ref3")	zapisanie na dysku Flash danych pod nową nazwą « **ref3** »
nErr=lib:libload("ref2")	skopiowanie « **ref2** » do « **ref3** »
nErr=lib:libSave("ref3")	

ZARZĄDZANIE BŁĘDAMI

Wszystkie instrukcje muszą zwracać wartość: **0**: bez błędów

nErr= lib:libLoad("data")

11: błąd ładowania = interfejs nie odnosi się do biblioteki

12: niepoprawne dane lub programy w bibliotece – ryzyko uszkodzenia pliku

nErr = lib:libLoad("data") lib:libSave("data") libDelete(“data”)

Niepoprawna forma nazwy

20: niepoprawny katalog główny

21: niepoprawna ścieżka

22: niepoprawna nazwa biblioteki

nErr = lib:libLoad("data") lib:libSave() lib:libSave("data")

30: błąd odczytu/zapisu

nErr = lib:libSave("data")

31: błąd zapisu = biblioteka istnieje(nadpisanie niemożliwe: najpierw skasuj starą bibliotekę)

BIBLIOTEKI – ĆWICZENIE

STÄUBLI

Ćwiczenie nr 13: Wykorzystanie bibliotek – **zadanie nr 7**:

Tworzenie bibliotek pozycji celem obsługi kilku miejsc układania części.

Zadanie 1:

Wykorzystaj program z ćwiczenia 8 (zadanie 3 – „*Zamiana miejsc*”);

Utwórz bibliotekę pozycji „**ref1**” zawierającą punkty **A**, **B** i **C** z ćwiczenia 11;

Upewnij się, że zmienne w bibliotece są publiczne;

Naucz pozycji w bibliotece „**ref1**” z wykorzystaniem odpowiedniego narzędzia;

Skopiuj bibliotekę „**ref1**” do „**ref2**”, “**ref3**”, “**ref4**”;

Naucz pozycji w bibliotece “**ref2**”, “**ref3**”, “**ref4**” – inne pozycje niż w „**ref1**”;

Zadeklaruj wykorzystanie punktów z biblioteki “**ref1**” w głównej aplikacji;

Zmień program tak aby korzystał z punktów z biblioteki “**ref1**”;

Wykonaj program.

Zadanie 2:

Zmodyfikuj program tak aby w zależności od stanu wejścia cyfrowego korzystał z punktów z “**ref1**”, “**ref2**”, “**ref3**” lub “**ref4**”.

TX-TS-RX-RS-TP/**CS8C**

TX2/**CS9**

- CZĘŚĆ 3 – PROGRAMISTA CS9

CALIBRATION - ADJUSTMENT

CALIBRATION VS ADJUSTMENT

- Dwie procedury są dostępne na kontrolerze CS8/C oraz CS9: **Calibration** i **Adjustment**
- **Calibration** (***F5 Rec*** - *recovery*) **–** kalibracja polega na przywróceniu położenia kątowego poszczególnych osi ramienia w sterowniku (w stopniach)
  - Ta pozycja jest mierzona za pomocą bezwzględnych enkoderów numerycznych firmy Heidenhain (po jednym na każdej osi)
  - Każda oś jest wyposażona w znacznik (stały i ruchomy), który jest bardzo blisko położenia geometrycznego zera
  - Odzyskiwanie to odbywa się kompletnie (wszystkie osie) lub częściowo utrata (1 lub 2 osi), może być przyczyną:
    - fizyczne odłączenie sprzężenia zwrotnego enkodera (typ kontrolera CS8)
    - lub utrata komunikacji na sprzężeniu zwrotnym enkodera podczas ruchu (typy kontrolerów CS8, CS8C lub CS9)
- **Adjustment** (***F1 Adj*** - *adjustment*) **–** korekta polegająca na wymuszeniu wspólnej pozycji do określonej wartości. Korekta taka jest wykonywana gdy:
  - Wykonana była ludzka interwencja polegająca na mechanicznym oddzieleniu enkodera, silnika od przekładni
  - Kolizja, która powoduje mechaniczną zmianę, której enkoder nie zauważa

RÓŻNE TYPY ENKODERÓW

- Każda pozycja jointowa jest obliczona przy użyciu dwóch informacji:
  - Ilość obrotów enkodera
  - Pozycja enkodera przy ostatnim obrocie

RX/**CS8 (stary zakres robotów)**

Liczenie obrotów (DAPS board)

6 x pozycja w jednym obrocie

Sercos bus (2 światłowody)

Numeryczne enkodery **single-turns**

Informacja zawarta w ramieniu:
- 6 x pozycja w jednym obrocie

TX-TS-RX-RS-TP/**CS8C**

TX2/CS9

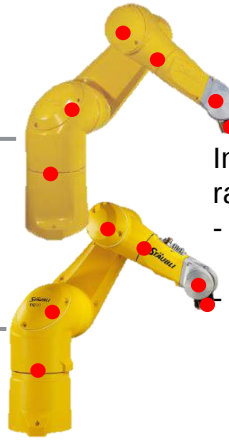
Pozycja absolutna (komplet)

Sercos bus (2 światłowody)

Pozycja absolutna (komplet)

EtherCAT bus (przewód miedziany)

Absolutne numeryczne enkodery **multi-turns**



Informacja zawarta w ramieniu:
- 6 x pozycja w jednym obrocie
- **6 x ilość obrotów**

KALKULACJA POZYCJI

- Dla kontrolera typu **CS8**, liczenie obrotów jest wykonywane przez moduł **DAPS** znajdującą się wewnątrz kontrolera
  - Ten moduł jest zasilany bateryjnie aby cały czas działał
  - W przypadku utraty zasilania lub ewentualnie rozłączenia między ramieniem a sterownikiem, CS8 nie rozpoznaje pozycji (jedna lub kilka osi)
  - **Calibration** – kalibracja pozwala na odzyskanie tych pozycji:
    - Odzyskiwanie to odbywa się poprzez umieszczenie danej osi lub wszystkich osi na znacznikach. Wymagana jest duża dokładność tego ustawienia w stosunku do jednego obrotu enkodera. Następnie automatyczna procedura wywołana z pilota zainicjuje liczenie obrotów licząc od tej pozycji

- Dla kontrolerów **CS8C** i **CS9**, liczenie to odbywa się w ramieniu robota, utrata pozycji ramienia ze sterownika może nastąpić jedynie w przypadku utraty łączności, wykorzystywanej do odzyskania tej pozycji podczas ruchów
  - Kalibracja pozwala kontrolerowi ponownie odzyskać pozycję ramienia. Informacja ta, pochodząca z ramienia jest kompletna. Procedura może być wykonana (z pilota) w dowolnej pozycji ramienia (**F5 Rec.**)

TX-TS-RX-RS-TP/**CS8C**

TX2/**CS9**

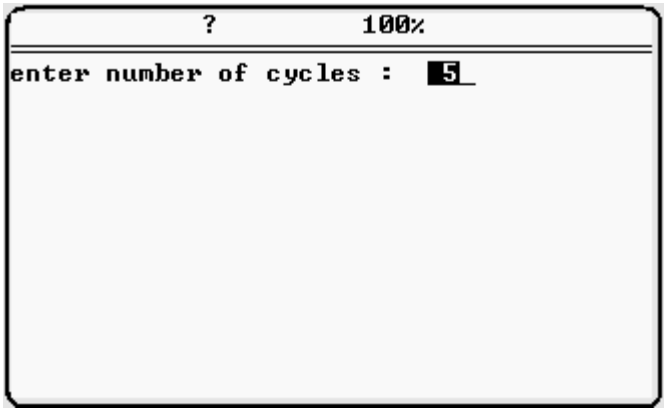
- CZĘŚĆ 3 – PROGRAMISTA CS9

HMI – INTERFEJS UŻYTKOWNIKA NA CS8C

EKRAN UŻYTKOWNIKA – USER PAGE

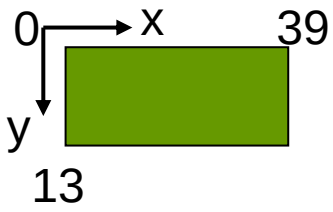
Wyświetlany po naciśnięciu **USER**

Ekran użytkownika pozwala na wyświetlanie komunikatów i wprowadzanie danych.



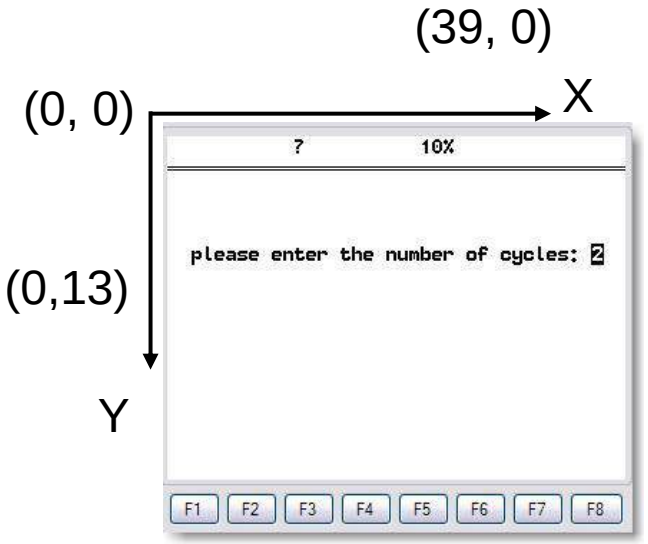
userPage() :	Wyświetlenie ekranu użytkownika
cls() :	Czyszczenie ekranu użytkownika
title(string) :	Nadaje tytuł ekranowi użytkownika

USER PAGE – WYŚWIETLANIE NAPISÓW



gotoxy(nX, nY)

Ustawia kursor na zadanej pozycji



put(num) / put(string)

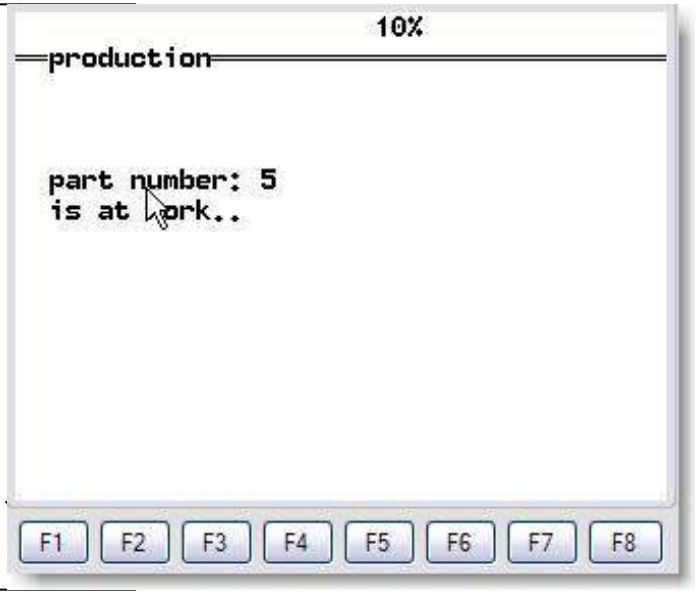
Wyświetla tekst lub wartość zmiennej numerycznej na aktualnej pozycji kursora. Po zakończeniu działania, kursor jest ustawiany za ostatnim znakiem.
(bez przejścia do następnej linii)

putln(num) / putln(string)

Tak jak put() lecz następuje przejście do kolejnej linii

Declaration:
num nIndex
string sMessage

cls()
title("Production")
gotoxy(2,3)
put("part number: ")
nIndex = 5
putln(nIndex)
sMessage=" is at work ..
putln(sMessage)



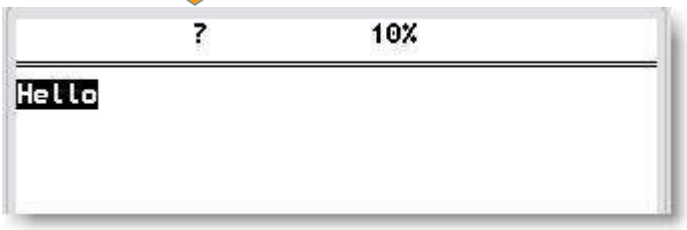
USER PAGE – WPROWADZANIE DANYCH

num get(num&) lub **num get(string&)**

Instrukcje wstrzymujące – czekają na potwierdzenie przyciskami

- “ENTER”
- “ESC”

Wyświetlany jest “?” w linii statusu, gdy aplikacja oczekuje na wprowadzenie danych.



num get()
Oczekuje na naciśnięcie przycisku i zwraca kod przypisany do niego.

num getKey() zwraca kod ostatnio naciśniętego przycisku. Zwraca -1 kiedy nie naciśnięto przycisku. Instrukcja nie wstrzymuje programu.

bool isKeyPressed(num) Zwraca TRUE jeżeli przycisk jest wciśnięty.

Declaration:
num nKey, num nValue
string sText, bool bPressed

sText = “Hello“
nKey = get(sText)

nValue = 5
nKey = get(nValue)

nKey = get()

nKey = getKey()

// 271 is the keycode of F1
bPressed = isKeyPressed(271)

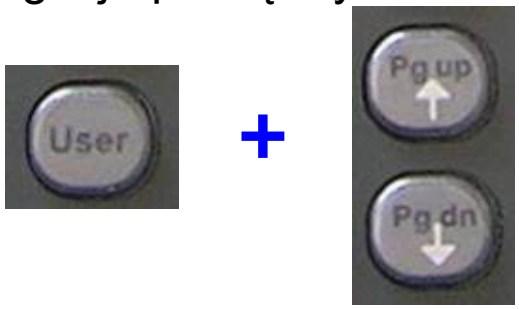
WIELE EKRANÓW: ZMIENNE EKRANOWE

- Zmienne **SCREEN** pozwalają na zarządzanie wieloma ekranami użytkownika
- Takie same instrukcje:

userPage(screen) ➔ **wyświetla ekran wybrany poprzez zmienną**
cls(screen)
title(screen,string)
gotoxy(screen,num,num)
put / putln(screen,num)
get(screen,num) lub get(screen,string)
…
userPage(scPage1)
title(scPage1, " Menu")
putln(scPage1,“Running …")
title(scPage2, " Settings")
…

```
  config
  trsf
-screen
   scPage1
   scPage2
-Programs
 +start
 +stop
```

Nawigacja pomiędzy ekranami:



KONWERSJA ZMIENNYCH NUM NA ZMIENNE STRING

Konwersja zmiennej numerycznej na tekstową z zaokrągleniem

message =toString("x.y",val)

val: konwertowana zmienna numeryczna

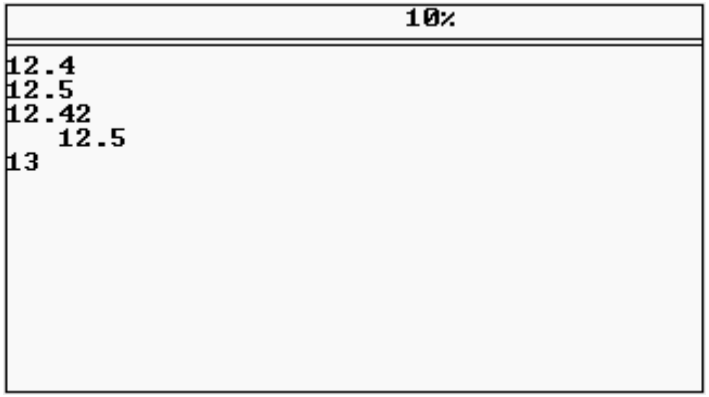
message: zmienna STRING rezultat konwersji

"x.y" : formatowanie x > y !!!!!!!!!

x: minimalna ilość znaków

y: numer miejsc po przecinku

```
putln(toString(“2.1“, 12.42))
putln(toString(“2.1“, 12.49))
putln(toString(“3.3“, 12.42))
putln(toString(“7.1“, 12.49))
putln(toString(“0.0“, 12.51))
```



USER PAGE – TIMER

num clock()

- Zwraca aktualną wartość zegara systemowego w sekundach;
- Dokładność w ms;
- Podczas uruchomienia kontrolera wartość ustawiana jest na 0.

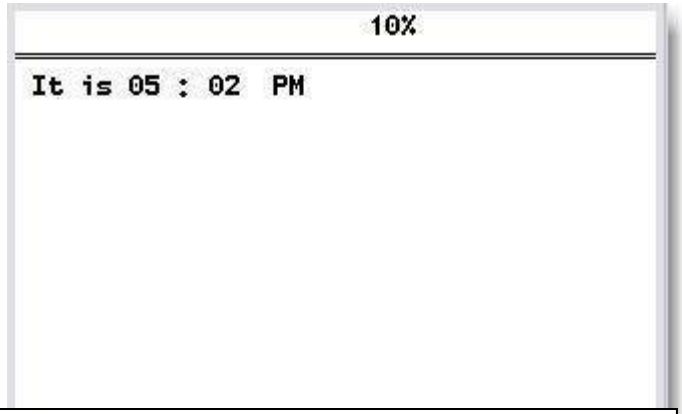
```
Declaration:
num nStartTime, nStopTime, nTotalTime

// eg. Cycle time measurement
nStartTime = clock()
...
...
nStopTime = clock()
nTotalTime = nStopTime - nStartTime
```

```
Declaration:
num nTime

// eg. Cycle time measurement
nTime = clock()
...
...
...
nTime = clock() - nTime
```

USER PAGE – DATA I CZAS

string getDate(string sFormat)

- Zwraca aktualną datę oraz godzinę ustawioną na kontrolerze
- Format jest określony parametrami. Wynik w formie zmiennej string:
  - %y 2 cyfry rok (00 – 99)
  - %Y 4 cyfry rok (np. 2008)
  - %m miesiąc (00 – 12)
  - %d dzień (00 – 31)
  - %H godzina (00 – 23)
  - %I godzina (01 – 12)
  - %p A.M. / P.M.
  - %M minuta (00 – 59)
  - %S sekunda (00 – 59)

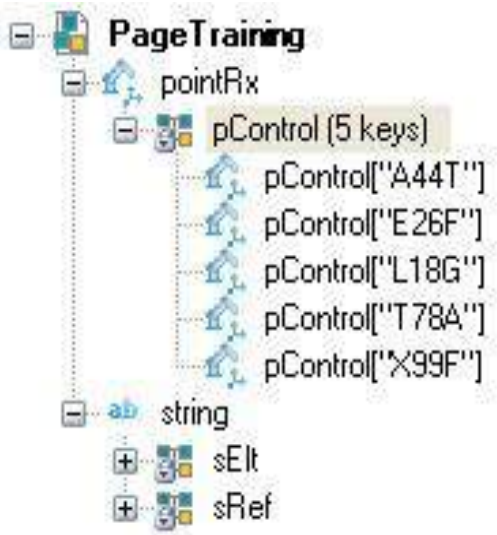


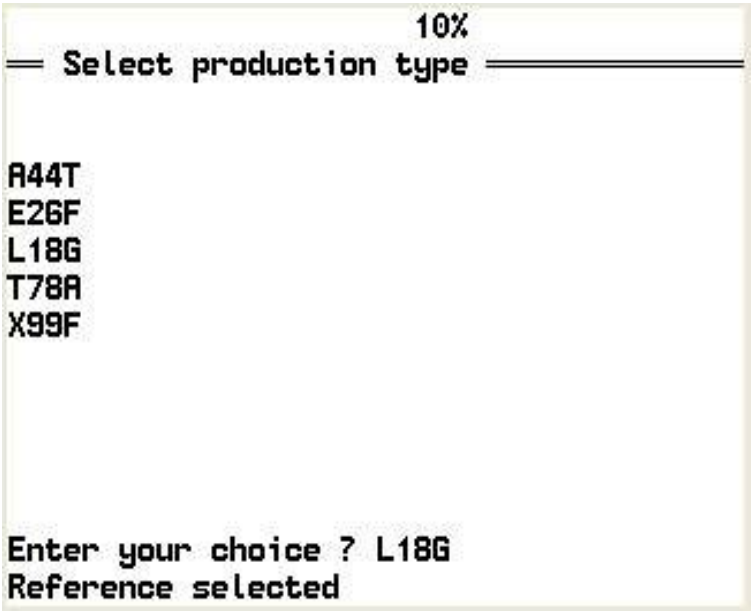
Declaration:
string sInput

sInput = "It is %I : %M %p"
putln(getDate(sInput))

PRZYKŁAD PROGRAMU EKRANU UŻYTKOWNIKA

```
//Wyświetla listę punktów w kolekcji
  userPage()
  cls()
  title(" Select production type ")
  gotoxy(0,2)
  sElt=first(pControl)
  while sElt!=""
    putln(sElt)
    sElt=next(pControl[sElt])
  endWhile
  gotoxy(0,12)
  put("Enter your choice ? ")
  get(sRef)
  putln(sRef)
  if isDefined(pControl[sRef])
    put("Reference selected")
  else
    put("Invalid reference")
    return
  endIf
...
```





TX-TS-RX-RS-TP/**CS8C**

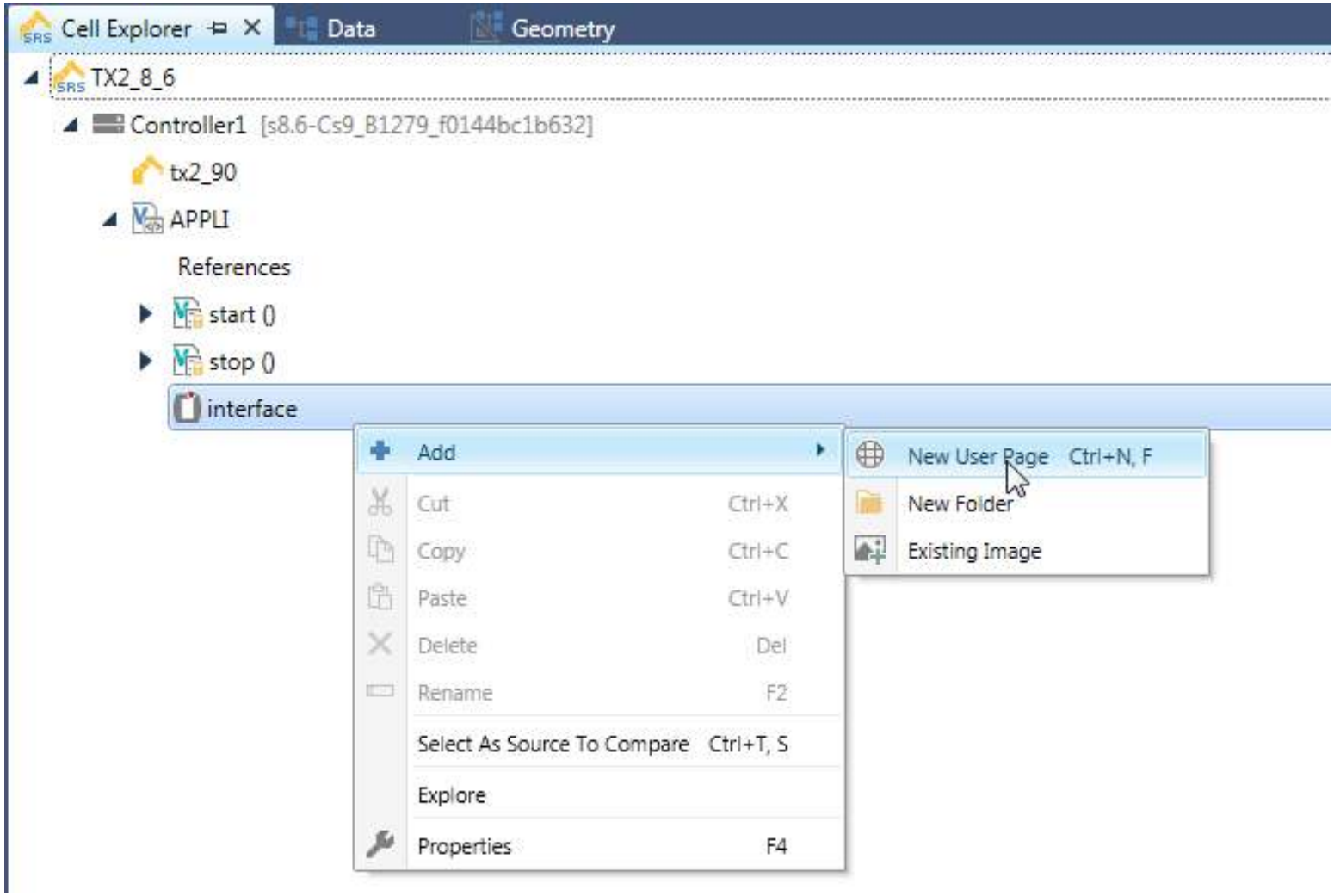
TX2/**CS9**

- CZĘŚĆ 3 – PROGRAMISTA CS9

HMI – INTERFEJS UŻYTKOWNIKA NA CS9

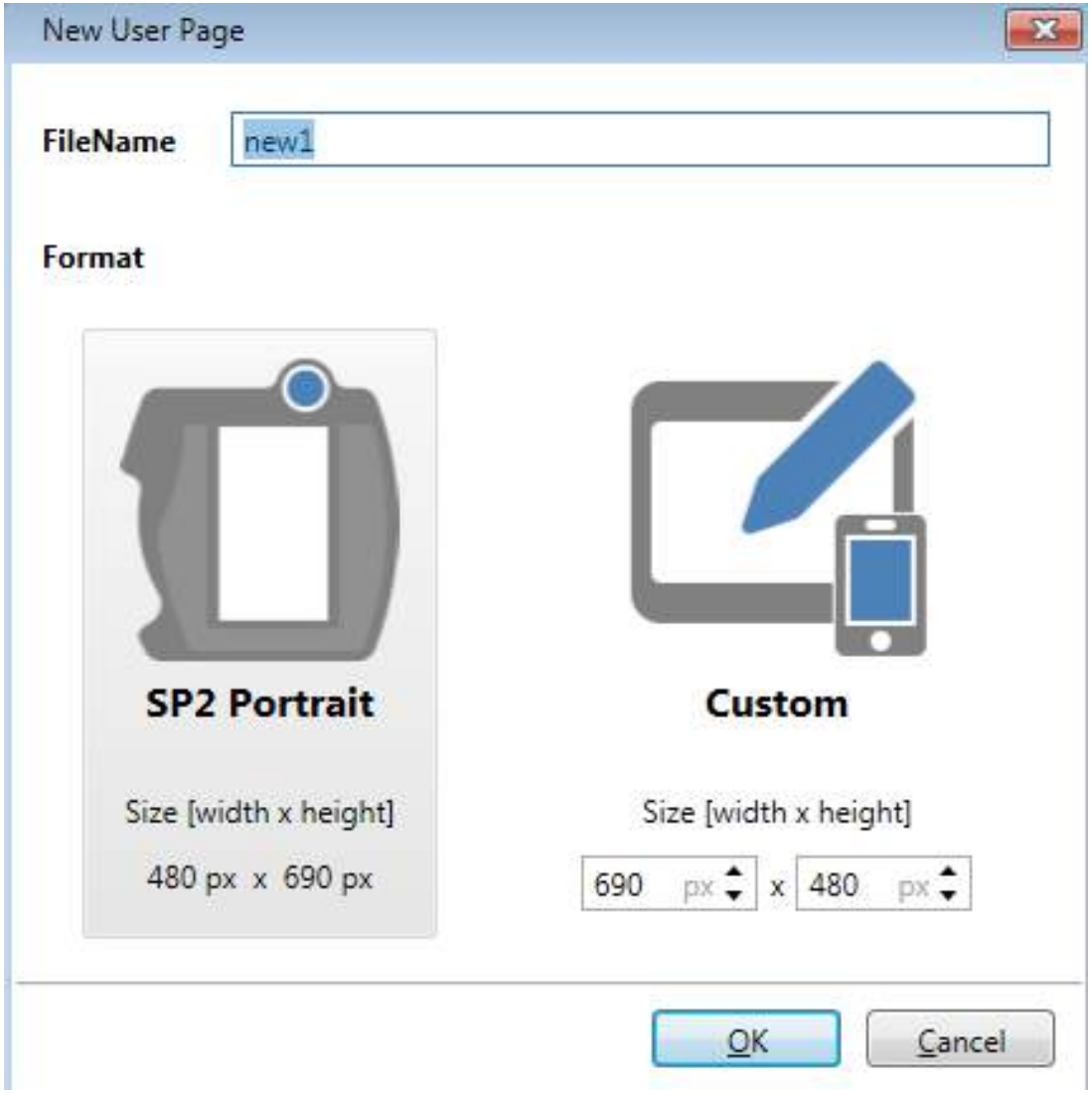
TWORZENIE EKRANÓW UŻYTKOWNIKA USER PAGES

- Nowa strona użytkownika jest tworzona z aplikacji VAL 3, w oknie celi



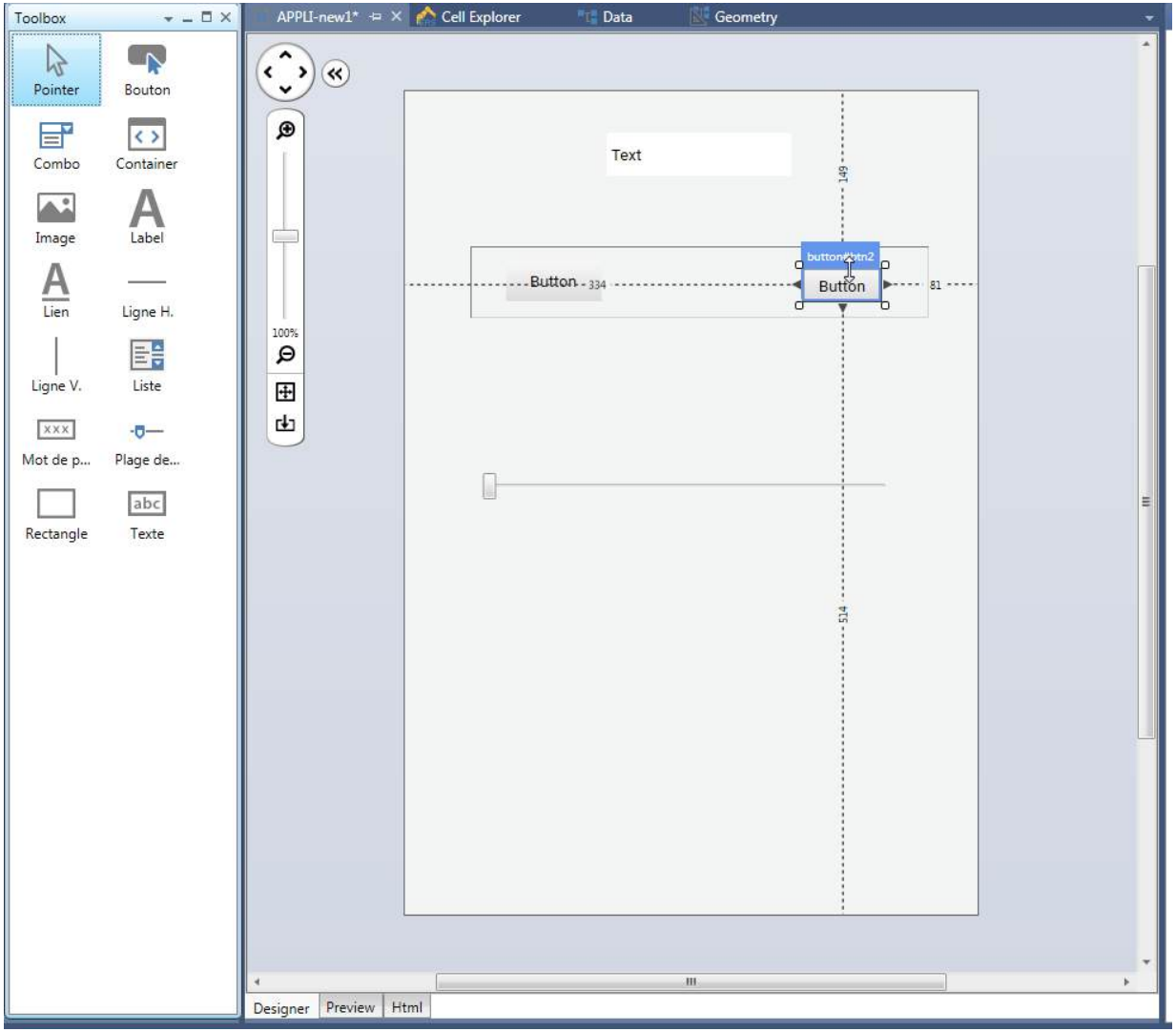
WYBÓR WYŚWIETLANIA

- Stronę można wyświetlić na pilocie SP2 lub na dowolnym innym ekranie przy użyciu przeglądarki internetowej



TOOL BOX

- Pozycjonowanie widgetów odbywa się graficznie z pola narzędzi (toolbox)
- Automatycznie proponowani są asystenci pozycjonowania



LINK Z VAL 3

- Wiązania można również edytować lub tworzyć za pomocą dedykowanych instrukcji VAL 3

Cellule | Vue 3D | demo_safety-interface

```
1   begin
2     //Displaying page 0
3     l_nPage=userPage("page0")
4     l_nPage=0
5     l_sCurrentPage="page0"
6     l_nStep=-1
7     sTodoTmp="nothing"
8     //Page 0 bindings
9     userPageBind(l_sCurrentPage,"btnEnd","innerHTML",sButtonText[0],1,"r",1)
10    userPageBind(l_sCurrentPage,"btnPause","innerHTML",sButtonText[1],1,"r",1)
11    userPageBind(l_sCurrentPage,"btnLaser","innerHTML",sButtonText[2],1,"r",1)
12    userPageBind(l_sCurrentPage,"btnLaser","style.backgroundColor",sButtonColor[4],1,"r",
13    userPageBind(l_sCurrentPage,"robStatus","innerHTML",sRobotStatus[0],1,"r",1)
14    userPageBind(l_sCurrentPage,"powStatus","innerHTML",sPowerStatus[0],1,"r",1)
15    userPageBind(l_sCurrentPage,"cyStatus","innerHTML",sCycleStatus[0],1,"r",1)
16    userPageBind(l_sCurrentPage,"todo","innerHTML",sTodo[0],1,"r",1)
17    userPageBind(l_sCurrentPage,"btnStart","style.backgroundColor",sButtonColor[0],1,"r",
18    userPageBind(l_sCurrentPage,"btnStop","style.backgroundColor",sButtonColor[1],1,"r",1
19    userPageBind(l_sCurrentPage,"btnEnd","style.backgroundColor",sButtonColor[2],1,"r",1)
20    userPageBind(l_sCurrentPage,"btnPause","style.backgroundColor",sButtonColor[3],1,"r",
21    userPageBindClick(l_sCurrentPage,"btnStart",bStartPressed,true)
22    userPageBindClick(l_sCurrentPage,"btnStop",bStopCy,true)
23    userPageBindClick(l_sCurrentPage,"btnEnd",bEndCy,true)
24    userPageBindClick(l_sCurrentPage,"btnPause",bPauseCy,true)
25    userPageBindClick(l_sCurrentPage,"btnLaser",nPage,1)
26
27    //Page 1 bindings
28    l_sCurrentPage="page1"
29    userPageBindClick(l_sCurrentPage,"btnBack",nPage,0)
30    userPageBind(l_sCurrentPage,"btnBack","innerHTML",sButtonText[2],1,"r",1)
31    userPageBind(l_sCurrentPage,"btnBack","style.backgroundColor",sButtonColor[4],1,"r",1
32    userPageBind(l_sCurrentPage,"safeStopZone","style.backgroundColor", sSafeZone[0],1,"r
33    userPageBind(l_sCurrentPage,"safeSpeedZone","style.backgroundColor", sSafeZone[1],1,"
34
35    l_sCurrentPage="page0"
36    while true
```

WYŚWIETLACZ

- Po przeniesieniu i uruchomieniu aplikacji, strony użytkownika będą wyświetlane na dołączonym ekranie (np. na pilocie SP2)

TX-TS-RX-RS-TP/**CS8C**

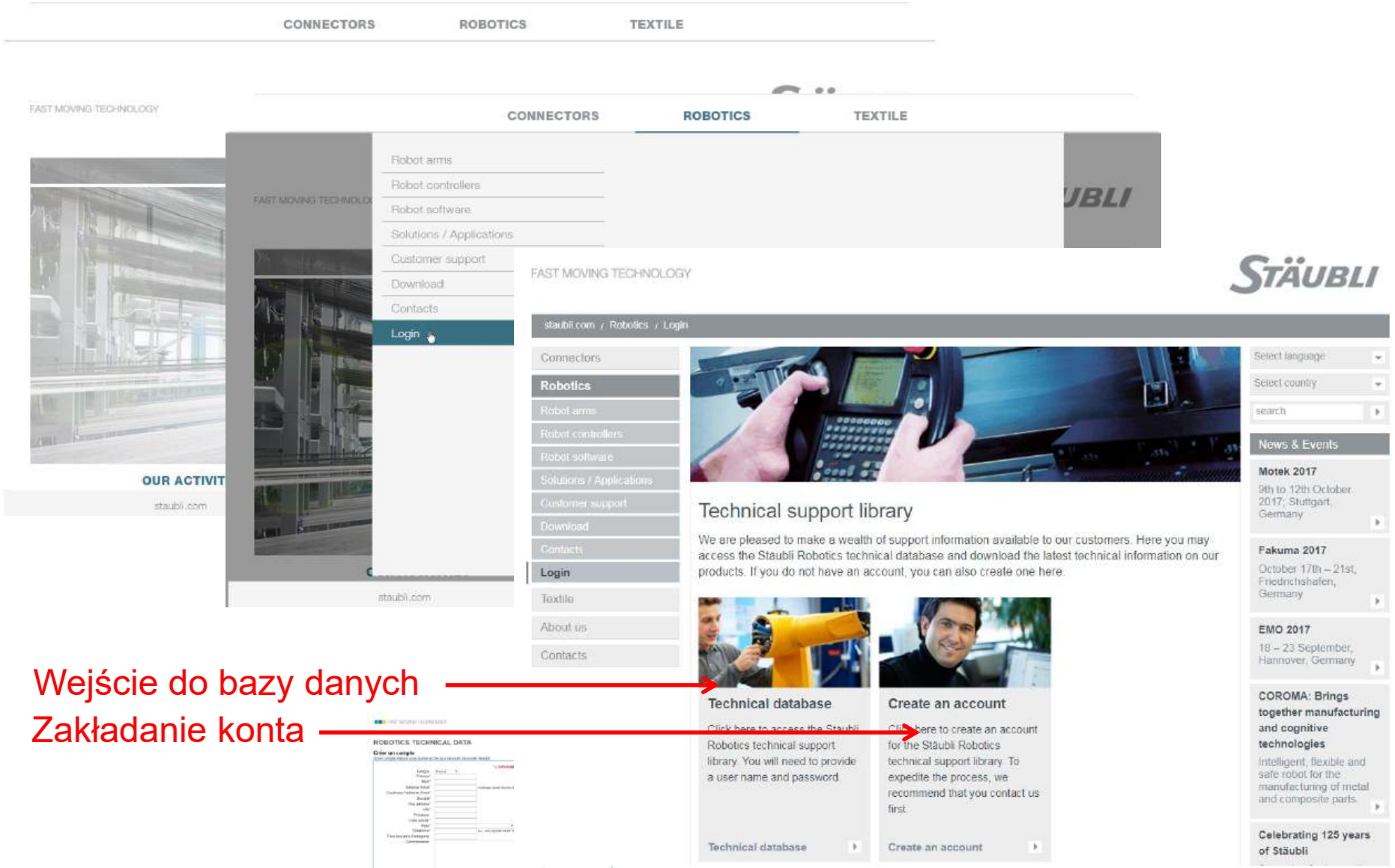
TX2/**CS9**

- CZĘŚĆ 3 – PROGRAMISTA CS9

DOSTĘP DO INTERNETOWEJ BAZY STÄUBLI

ZAKŁADANIE KONTA

http://www.staubli.com

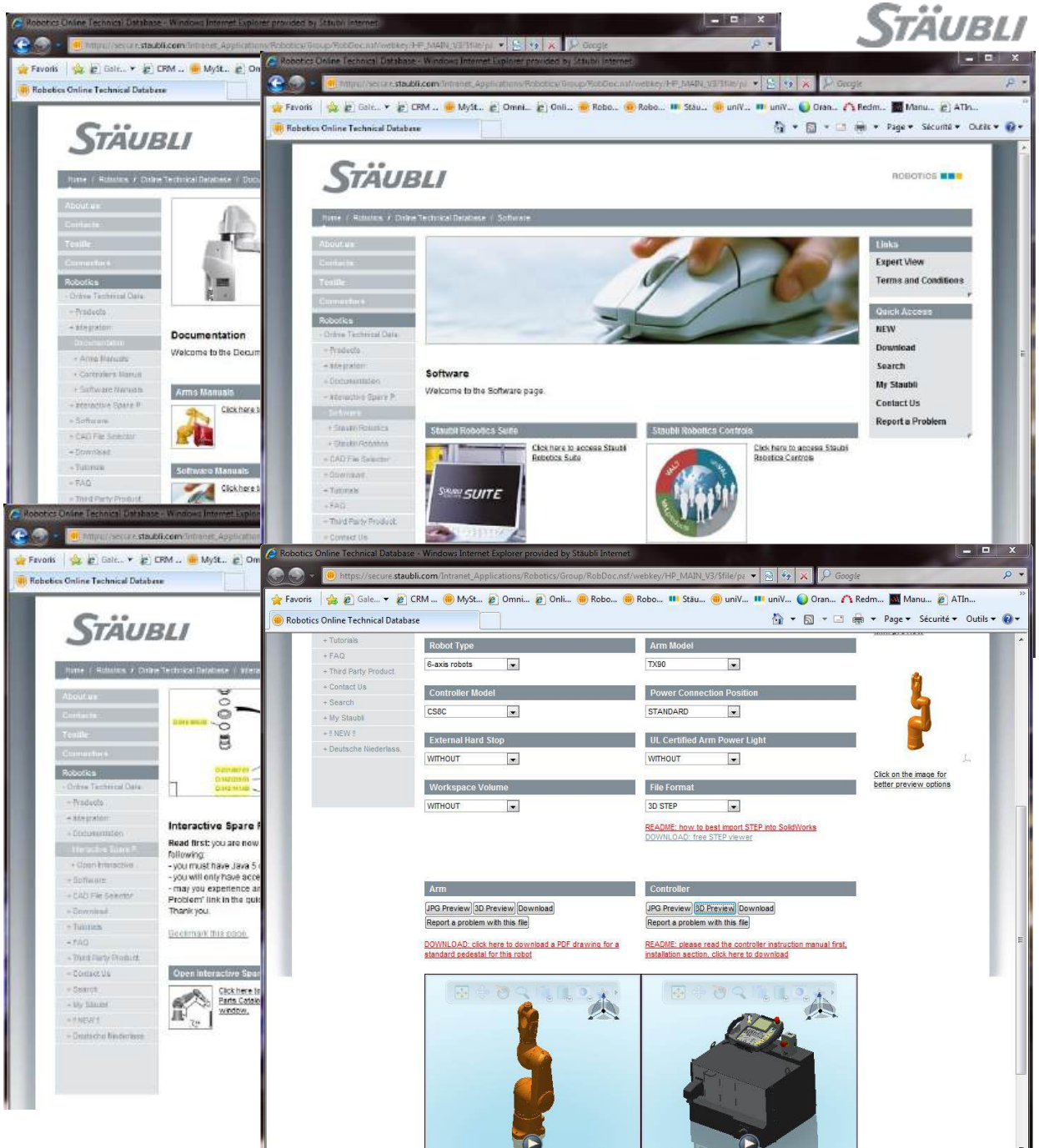


Wejście do bazy danych

Zakładanie konta

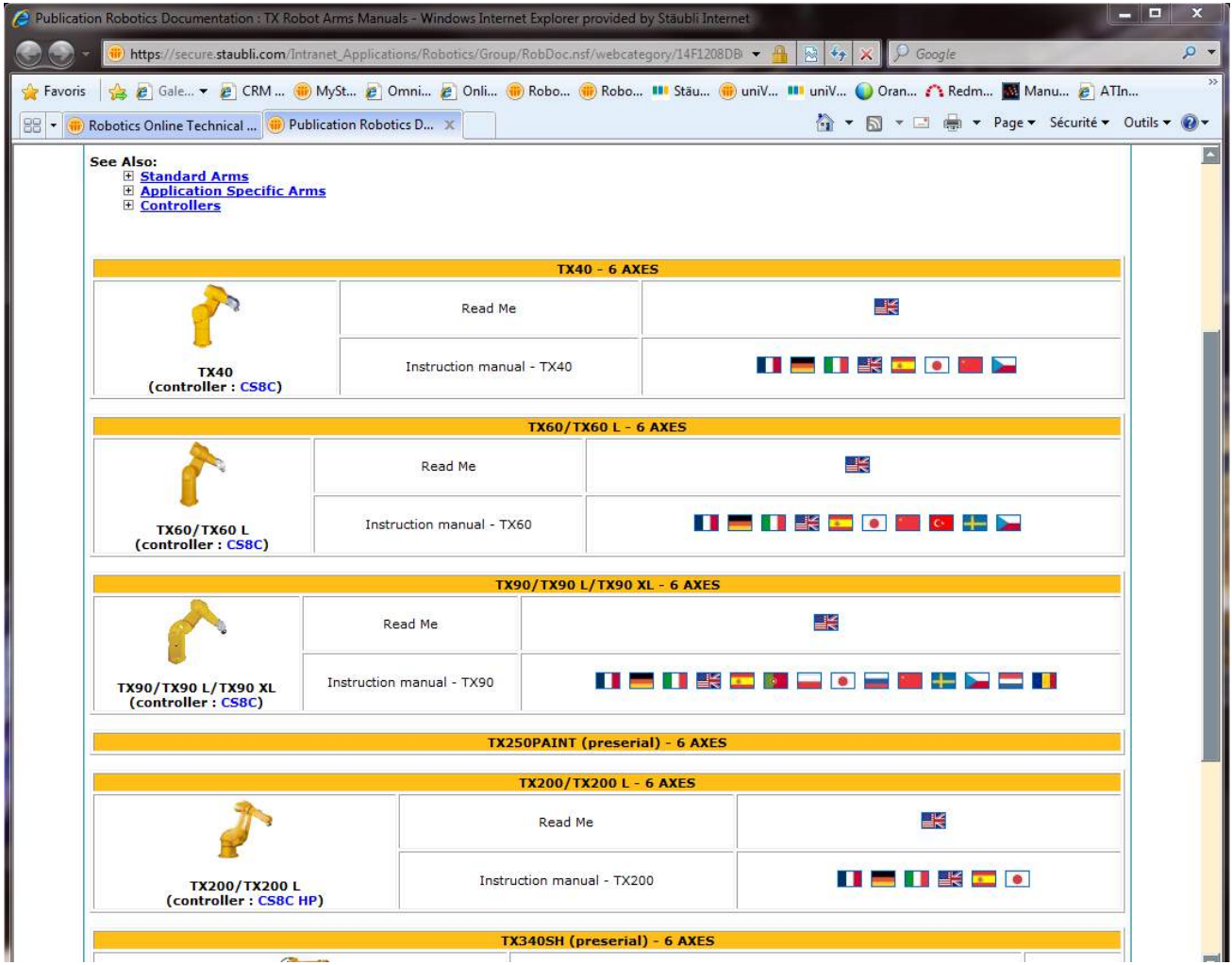
ZAWARTOŚĆ

- Dokumentacja
- Pliki CAD
- Katalog części zamiennych
- Software download
- Tutoriale
- i wiele więcej …



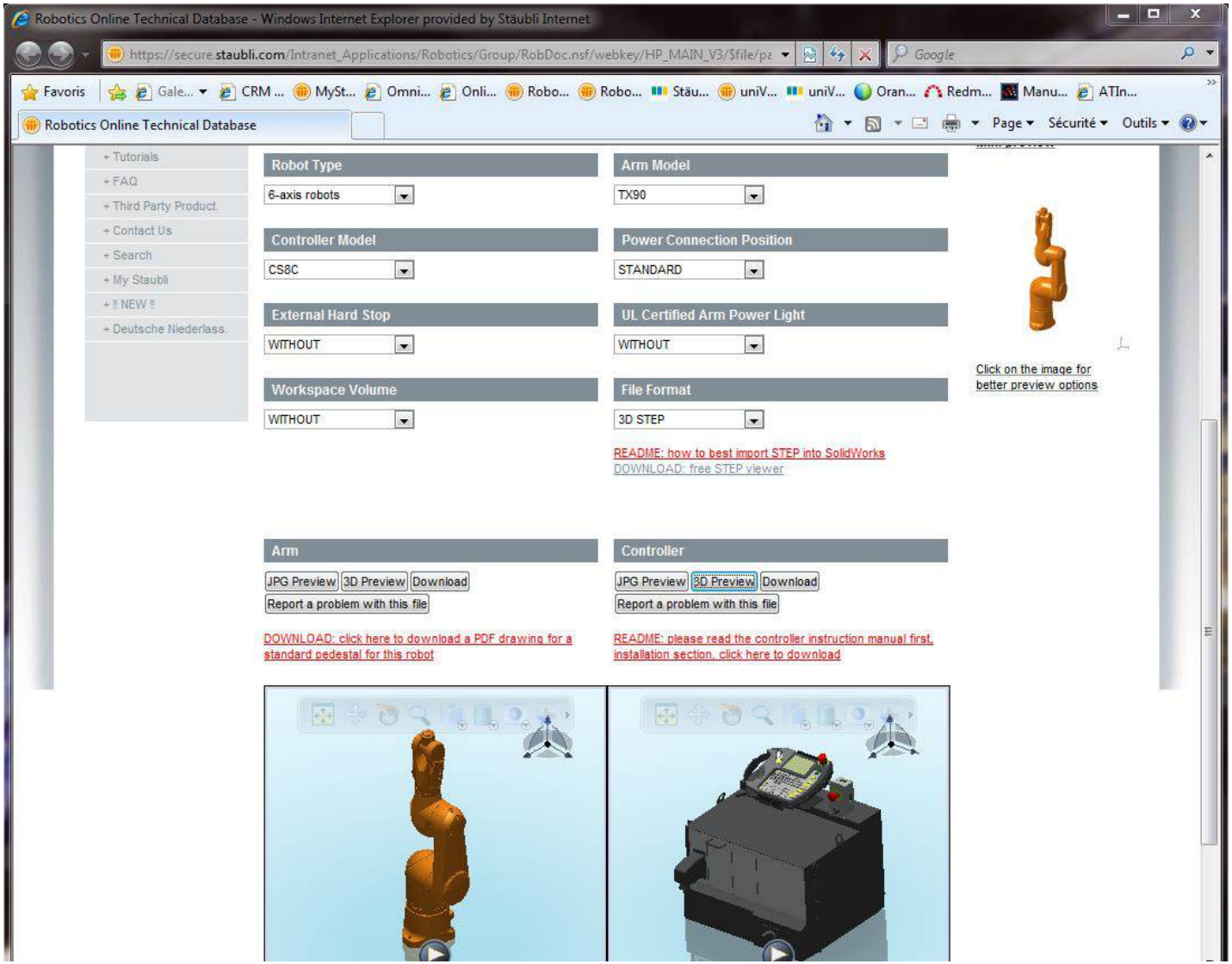
DOKUMENTACJA

- Wybór rodziny robotów
- Wybór modelu
- Wybór języka
- Manual download



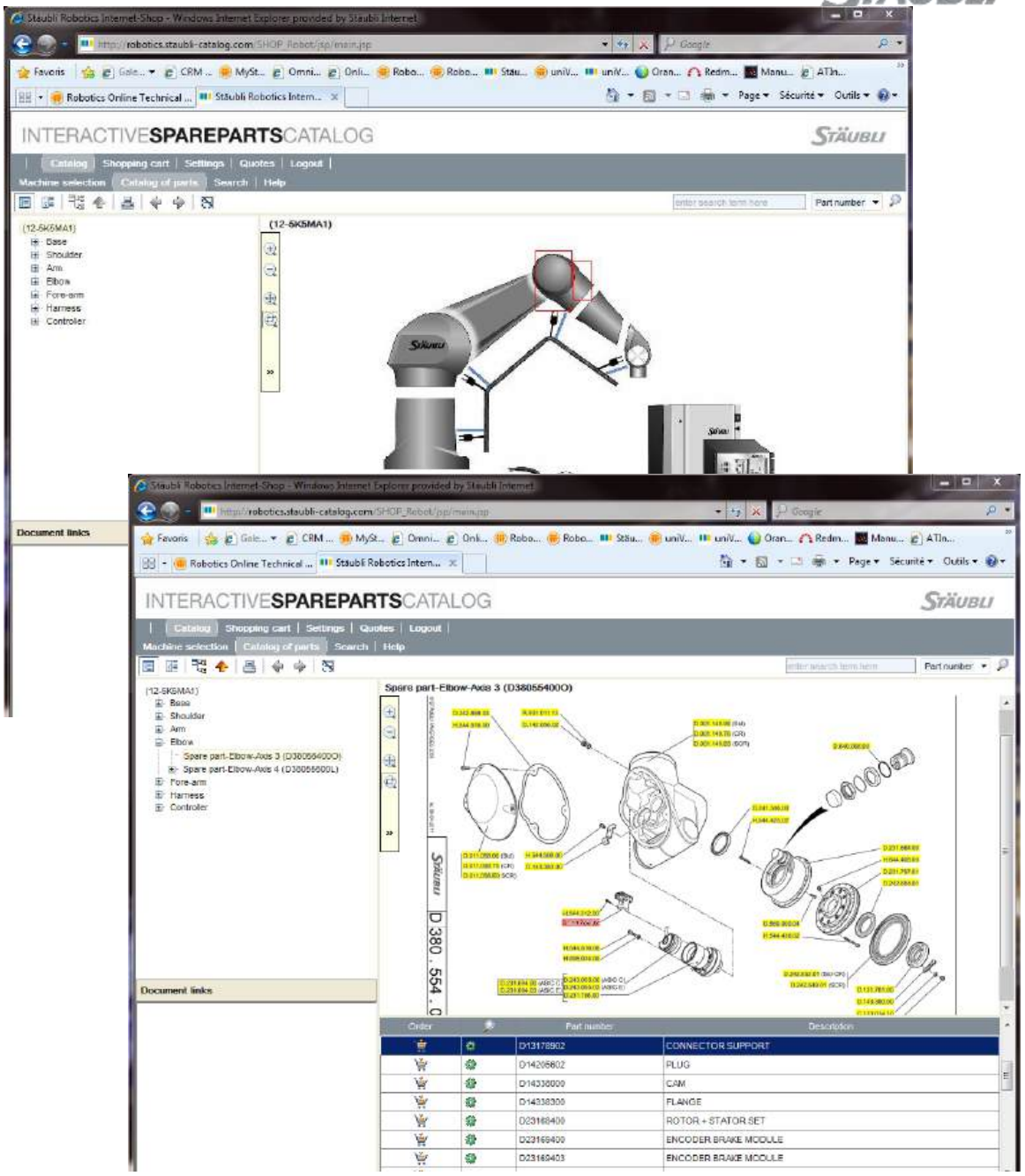
KONFIGUROWALNE PLIKI CAD

- Wybór robota
- Wybór opcji
- Podgląd 3D
- Wybór formatu
- Pobieranie pliku ramienia
- Pobieranie pliku kontrolera



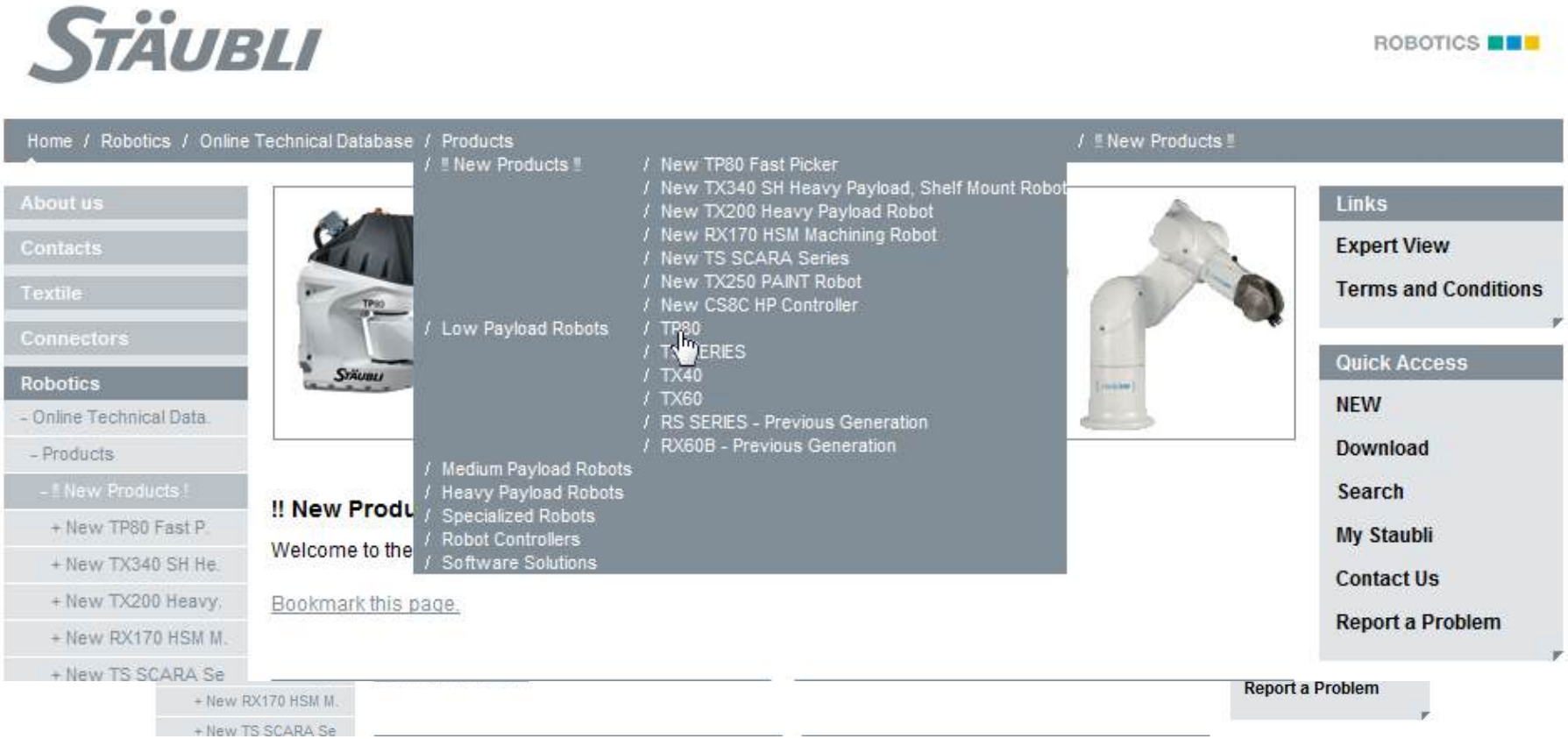
KATALOG CZĘŚCI ZAMIENNYCH

- Wyszukiwanie po numerze seryjnym
- Graficzny wybór korpusu robota
- Graficzny wybór części zamiennych
- Dodawanie części do koszyka
- Bezpośrednie wysyłanie zapytania o ofertę do Stäubli



INTERAKTYWNA MAPA STRONY

- Rozwijane menu, gdy mysz przechodzi nad sekcjami i podsekcjach
- Bezpośredni dostęp na stronie



STRONA « MY STÄUBLI »

- Dostosowana strona zapewnia szybki dostęp do ulubionych stron
- Podczas nawigacji link na każdej stronie pozwala dodać go jako zakładkę