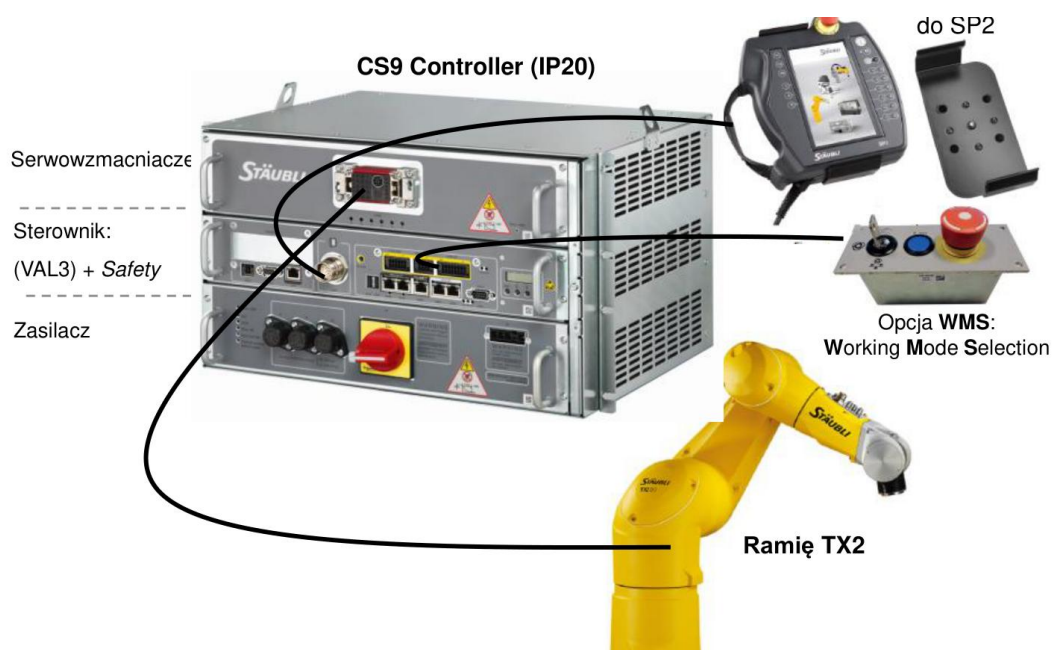


LABORATORIUM Z PROGRAMOWANIA ROBOTÓW

Stäubli TX2 / CS9



Elementy stanowiska: kontroler CS9, ramię TX2, teach pendant SP2 i WMS.

Ważne przypomnienie przed rozpoczęciem pracy

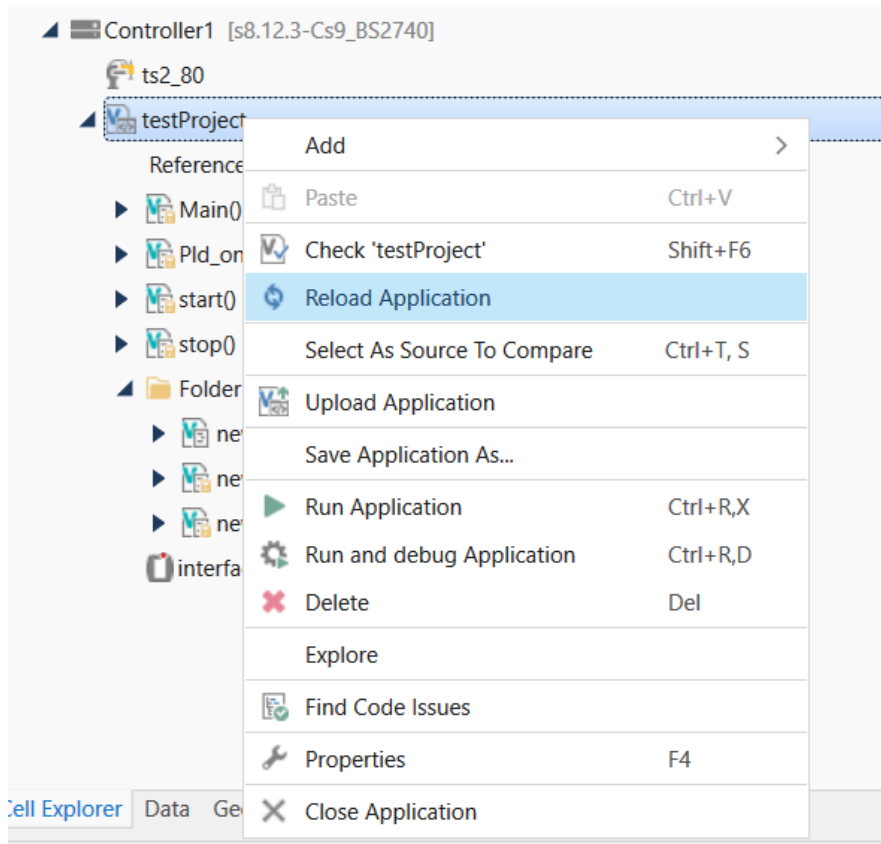


WAŻNE — zapisuj wszystko na panelu

- Jeżeli przy ikonie zapisu pojawia się gwiazdka (*), zmiany nie zostały zapisane. **Po każdej zmianie punktów, zmiennych i programu użyj Zapisz.**
- Zapisuj na bieżąco podczas pracy na SP2, szczególnie po nauce punktów i po edycji programu. **Nie zakładaj, że kontroler zapisał zmiany automatycznie.**
- Po transferze plików/projektu w jedną lub drugą stronę (PC ↔ kontroler) **zawsze wykonaj Reload Application / Reload Project przed dalszą pracą.**

Ikona dyskietki z gwiazdką oznacza niezapisane zmiany na panelu operatorskim.

Po przestaniu aplikacji/projektu z PC do kontrolera lub z kontrolera do PC wykonaj **Reload Application / Reload Project**, aby odświeżyć projekt przed dalszą pracą.



Przykład w Robotics Suite: po transferze użyj polecenia „Reload Application” dla aktywnej aplikacji/projektu.

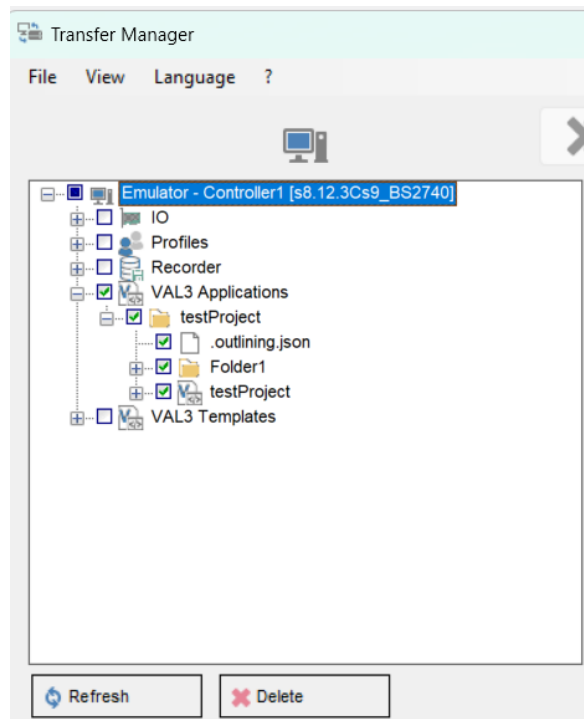
Transfer Manager — zasady bezpiecznego transferu

W Transfer Manager przed rozpoczęciem kopiowania zawsze sprawdź, które elementy są zaznaczone. Na laboratorium przenosimy tylko własny projekt/aplikację albo własne programy — nie konfigurację kontrolera.

UWAGA — nie przenoś ustawień kontrolera

- Zaznaczaj i przenoś wyłącznie swój projekt/aplikację. **Przed transferem sprawdź zaznaczenia w drzewie.**
- Jeżeli chcesz przenosić tylko wybrane programy, umieść je wcześniej w osobnym folderze. **W przeciwnym razie może zostać przeniesiony komplet programów aplikacji naraz.**
- Pod żadnym pozorem nie zaznaczaj ani nie przenoś gałęzi **I/O oraz Profiles**.
- Po każdym transferze między PC i kontrolerem **wykonaj Reload Application / Reload Project przed dalszą pracą.**

Na poniższym przykładzie w Transfer Manager zaznaczona jest aplikacja testProject w gałęzi VAL3 Applications. Gałęzie I/O i Profiles pozostają niezaznaczone — tak powinno być podczas pracy laboratoryjnej.



Transfer Manager — przed kopiowaniem sprawdź zaznaczenia. Przenoś tylko własną aplikację/projekt lub przygotowany folder z programami; I/O i Profiles pozostaw niezaznaczone.

1. Stanowisko i bezpieczeństwo

- **TX2:** 6-osiowe ramię przemysłowe; osie J1–J6 prowadzą od podstawy do flanszy.
- **CS9:** kontroler zawierający sterownik VAL3, Safety, serwowzmacniacze i zasilacz.
- **SP2:** teach pendant do sterowania ręcznego, nauki punktów i obsługi programu.
- **WMS:** Working Mode Selection – wybór trybu pracy; przełącznik powinien znajdować się poza celą.
- **TCP:** Tool Center Point – punkt roboczy narzędzia; dla domyślnego Flange leży w środku flanszy osi 6.
-



Teach pendant SP2 – E-STOP, Servo ON/OFF, Move/Hold, przyciski ruchu, Deadman i nawigacja.

Ważne: Pierwszy cykl wykonuj z małą prędkością. E-STOP stosuj awaryjnie; Move/Hold daje łagodne zatrzymanie bez wyłączenia serwonapędów.

Tryby pracy

- **Automatyczny:** cela zamknięta, ruch z programu; lokalnie lub zdalnie przez PLC; możliwe duże prędkości.
- **Ręczny:** operator steruje z SP2; służy do nauki punktów, JOG i dojazdów; prędkość ograniczona do 250 mm/s.

1

Załączenie głównego wyłącznika

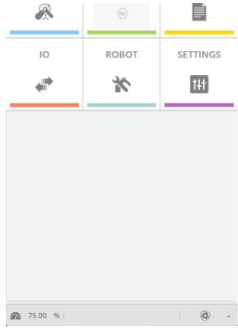


2

Poczekaj na wyświetlenie się głównego menu



LUB



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

Uruchomienie kontrolera: główny wyłącznik i oczekiwanie na menu/HMI.

2. Uruchamianie i załączanie napędów

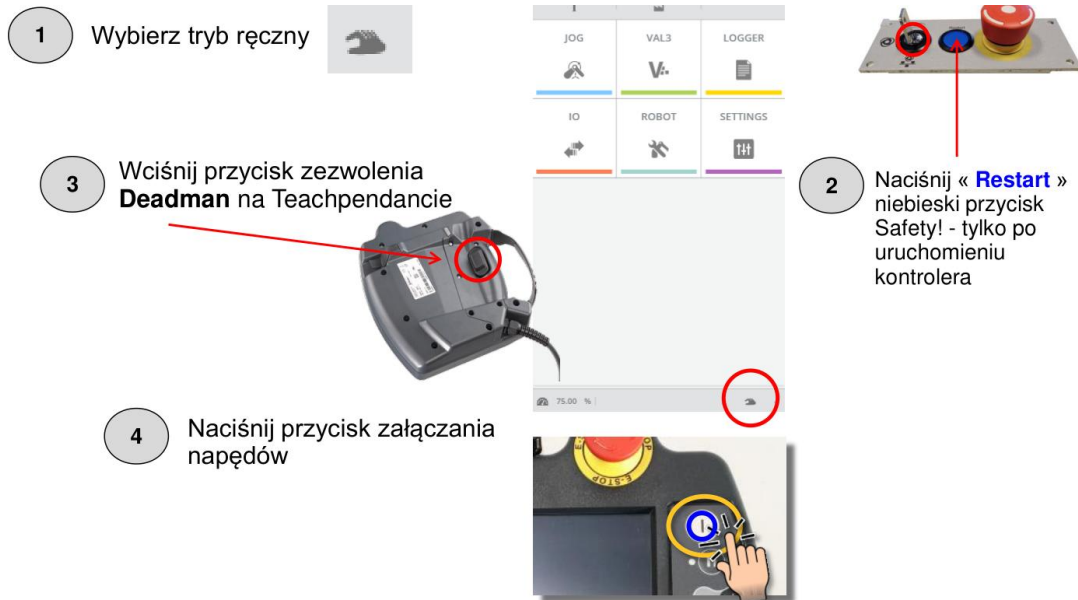
Tryb automatyczny

1. Sprawdź brak aktywnego E-STOP.
2. Wybierz tryb automatyczny lokalny lub zdalny.
3. Po uruchomieniu kontrolera w razie potrzeby wykonaj Restart Safety.
4. Lokalnie: załącz serwonapędy. Zdalnie: napędy może załączyć urządzenie zewnętrzne, np. PLC.

Tryb ręczny

1. Wybierz tryb ręczny.
2. Po starcie kontrolera wykonaj Restart Safety, jeżeli jest wymagany.
3. Wciśnij i trzymaj Deadman na SP2.
4. Naciśnij przycisk załączania napędów.

Ważne: Po puszczeniu Deadmana napędy w trybie ręcznym zostają wyłączone.

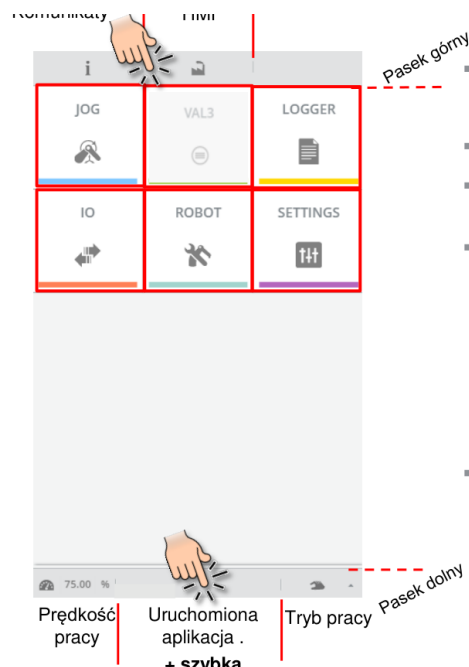


UWAGA: Napędy zostaną wyłączone po puszczeniu przycisku zezwolenia „Deadman”

Załączanie serwonapędów w trybie ręcznym.

3. Menu, profile, komunikaty i logi

- **JOG:** ruch ręczny, nauka i dojazd do punktów.
- **VAL3:** obsługa i edycja aplikacji.
- **LOGGER:** logi błędów.
- **I/O:** wejścia/wyjścia oraz sygnały bezpieczeństwa.
- **ROBOT:** kalibracja, recovery, phasing, zwalnianie hamulców, recorder, safety.
- **SETTINGS:** sieć, język, data/godzina i profile użytkownika.



- 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)



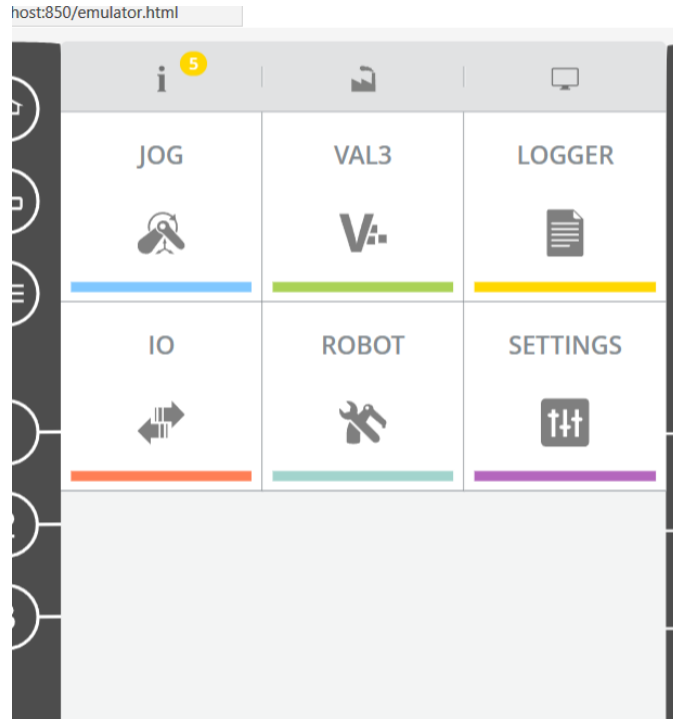
Strona startowa CS9 i najważniejsze kafelki.

Uruchamianie aplikacji z poziomu VAL3

W Stäubli Robotics Suite (SRS 2022) zarządzanie aplikacjami odbywa się w obszarze VAL3 application management. Na panelu SP2 uruchomienie przygotowanej aplikacji wykonuje się z poziomu okna VAL3: najpierw wybiera się moduł VAL3, następnie ładuje aplikację do pamięci roboczej i na końcu uruchamia ją przyciskiem Play.

Krok 1. Wejść do okna VAL3 z ekranu głównego panelu SP2.

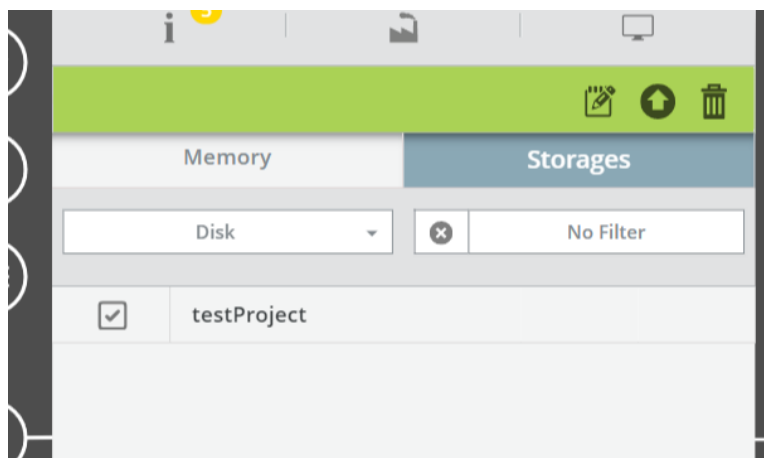
Na stronie startowej wybierz kafelek VAL3. To tutaj obsługuje się aplikacje, programy i dane związane z VAL3.



Ekran główny SP2 – wybór modułu VAL3.

Krok 2. Załaduj aplikację z listy, używając ikony strzałki w górę.

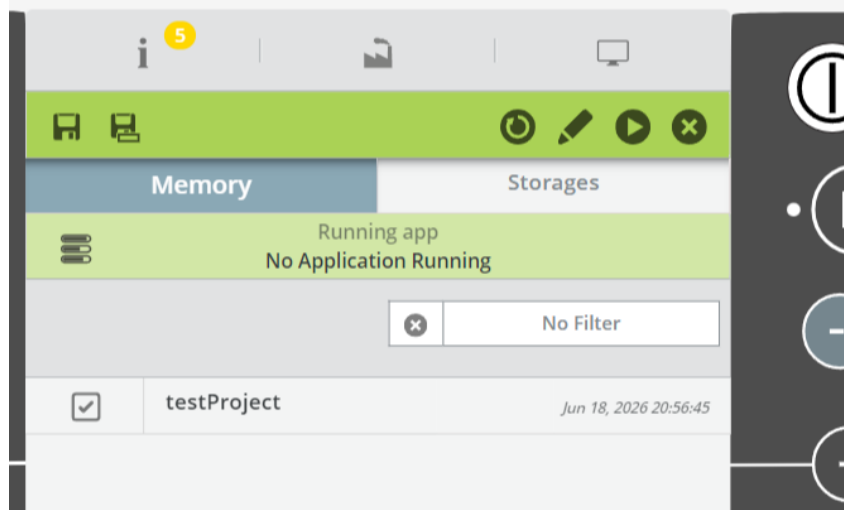
W oknie VAL3 odszukaj swoją aplikację (np. testProject), zaznacz ją i naciśnij ikonę strzałki w górę w górnym pasku. Jeżeli aplikacja była wcześniej przenoszona pomiędzy PC i kontrolerem, najpierw wykonaj Reload Application / Reload Project.



Okno VAL3 – zaznaczenie aplikacji i załadowanie jej ikoną strzałki w górę.

Krok 3. Uruchom aplikację przyciskiem Play.

Po załadowaniu aplikacji użyj przycisku Play (trójkąt w kółku) w górnym pasku. W tym samym miejscu można później zatrzymać aplikację lub sprawdzić, czy w polu Running app pojawia się uruchomiony program.



Okno VAL3 – uruchomienie aplikacji przyciskiem Play.

Wskazówki praktyczne

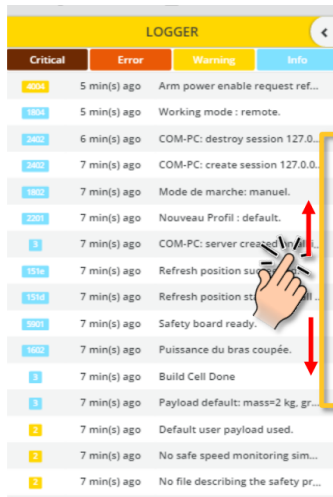
- Przed uruchomieniem aplikacji upewnij się, że projekt i zmienne są zapisane – ikona dyskiety z gwiazdką oznacza niezapisane zmiany.
- Jeżeli aplikacja nie chce się uruchomić po transferze z SRS, wykonaj Reload Application / Reload Project, a następnie powtórz ładowanie i uruchomienie.
- Dobrą praktyką w SRS 2022 jest wcześniejsze użycie polecenia Check Syntax, aby wykryć błędy aplikacji jeszcze przed jej uruchomieniem.

Profile użytkownika

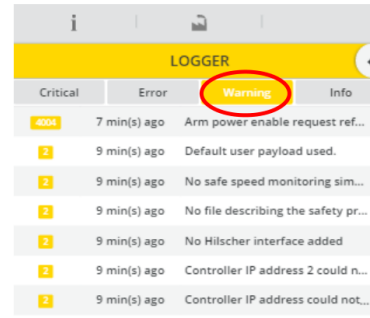
- Current profile = aktywny profil; Starting profile = profil startowy.
- Profile domyślne: default (bez hasła), maintenance i staubli (chronione hasłem).

Komunikaty i LOGGER

- Kategorie komunikatów: Critical, Error, Warning, Info.
- Logi są sortowane wg daty i czasu; można je filtrować i otwierać pełny opis.
- Eksport: podłącz USB FAT/FAT32 → LOGGER → menu boczne → USB0 → Export.



- 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**

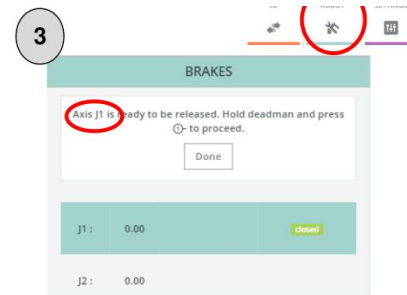
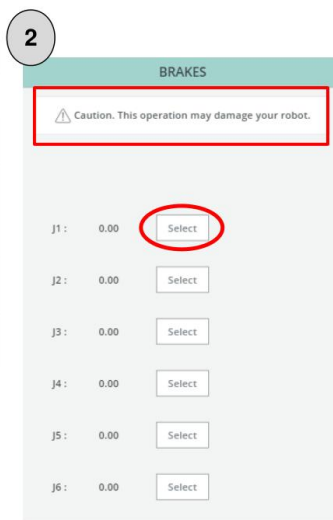
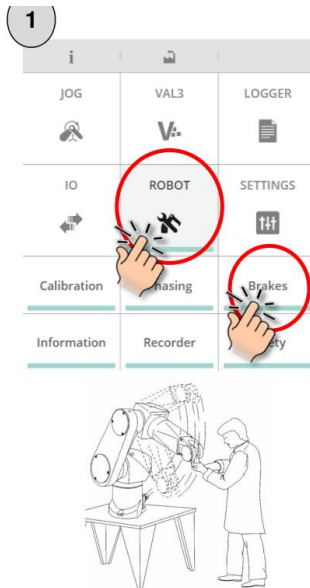


- Przewijanie listy za pomocą palca
- Wyświetlenie pełnego komunikatu po kliknięciu palcem na komunikacie

LOGGER – lista i filtrowanie komunikatów.

4. Zwolnienie hamulców

- Brake release jest dostępny w menu ROBOT.
- Przed zwolnieniem hamulców należy podtrzymać ramię, aby nie opadło pod wpływem ciężaru.
- Opcja R.B.R. zmniejsza prędkość opadania; może działać przy wyłączonym kontrolerze z zewnętrznym zasilaniem 24 V.



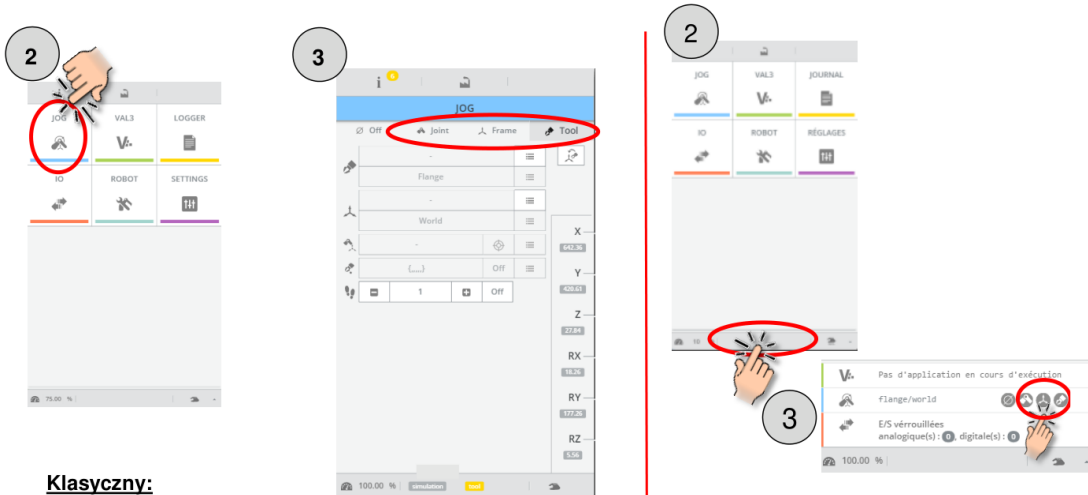
!!! Przytrzymaj ramię zanim zwolnisz hamulce !!!

Zwalnianie hamulców – kolejność i ostrzeżenie o podtrzymaniu ramienia.

5. Ręczne poruszanie robotem – JOG

1. Ustaw tryb ręczny.
2. Wejdź do JOG lub użyj szybkiego menu.
3. Wybierz ruch osiowy (Joint), ruch w układzie współrzędnych albo ruch względem narzędzia (Tool).
4. Sprawdź wybrane narzędzie i układ współrzędnych.
5. Trzymając Deadman, używaj przycisków +/-.

2 sposoby wyboru trybu ruchu



Klasyczny:

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

Szybki dostęp z dowolnego miejsca:

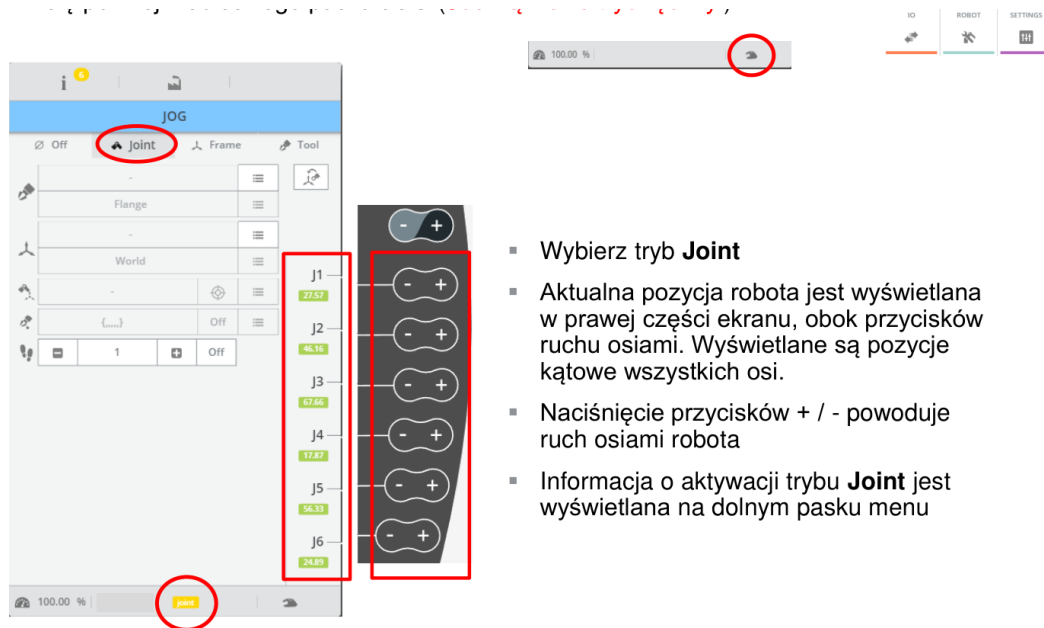
Środek dolnego paska menu

- Szybki wybór rodzaju ruchu bez zmiany menu

Wybór rodzaju ruchu: klasycznie w JOG albo z szybkiego menu.

JOINT

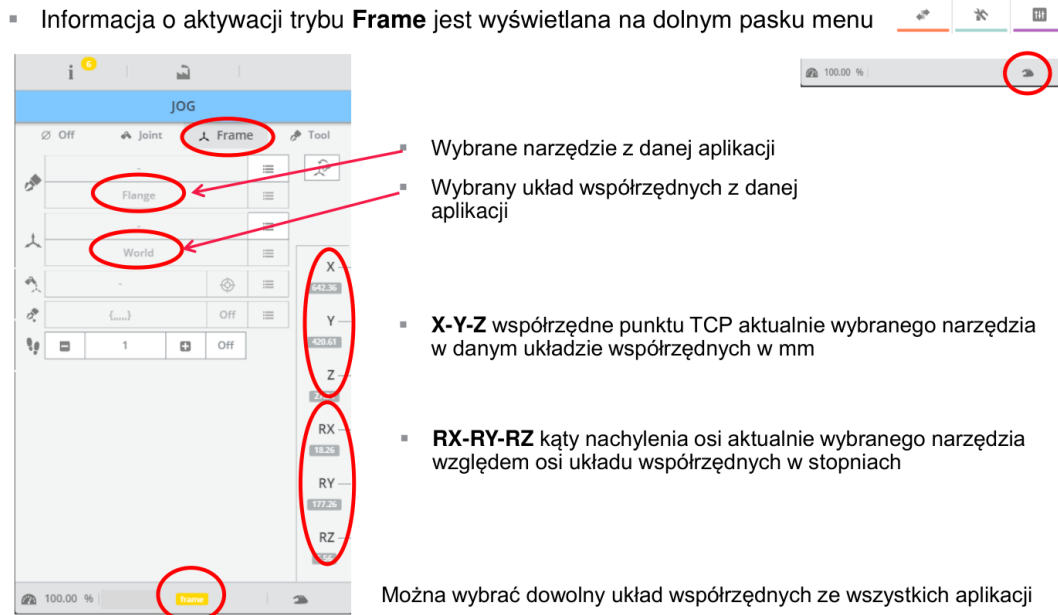
- Sterujesz bezpośrednio osiami J1–J6.
- Na ekranie wyświetlane są pozycje kątowe osi.
- Przyciski +/- wywołują ruch wybranej osi.



JOINT – ręczne sterowanie osiami J1–J6.

RUCH W UKŁADZIE WSPÓŁRZĘDNYCH

- World to domyślny, stały układ kartezjański związany z podstawą robota.
- X, Y, Z opisują położenie TCP w mm; RX, RY, RZ opisują orientację w stopniach.
- Sterownik dobiera ruch osi tak, aby TCP przemieszczał się względem wybranego układu.

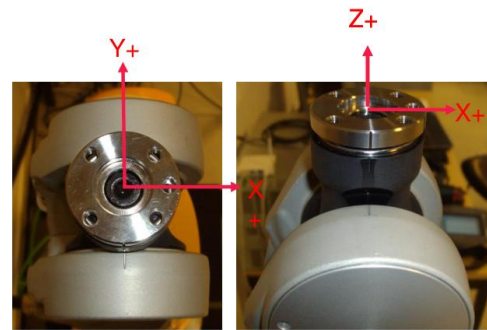
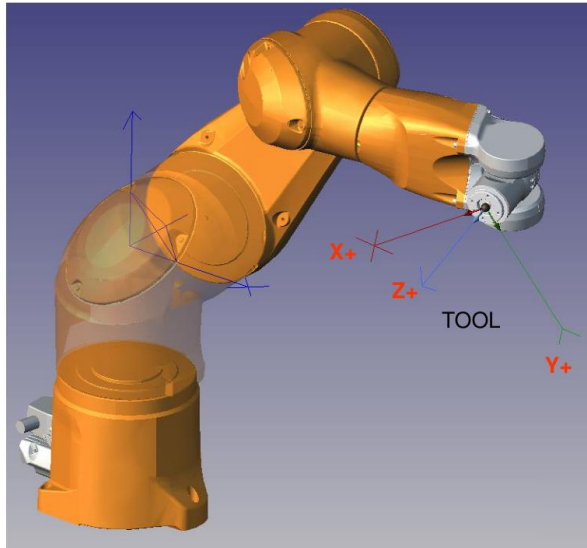


Ruch w układzie współrzędnych – wybór narzędzia/układu i podgląd X/Y/Z oraz RX/RY/RZ.

TOOL

- Ruch X/Y/Z odbywa się względem osi aktualnie wybranego narzędzia.
- Układ jest ruchomy; X i Y zmieniają kierunek wraz z orientacją osi 6.
- Oś Z narzędzia „wychodzi” z flanszy/TCP.

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

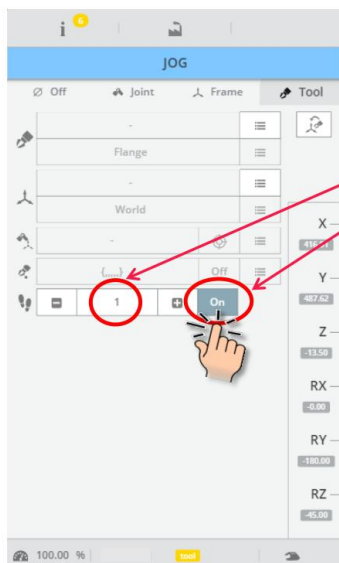
TOOL – poruszanie TCP względem osi narzędzia.

Ruch krokowy i Align

- Ruch krokowy działa dla ruchu osiowego (Joint), ruchu w układzie współrzędnych i ruchu względem narzędzia (Tool). Ustaw wartość kroku, włącz tryb, wykonaj ruch, potem wyłącz.
- Przykład: 10 mm dla X/Y/Z albo 10° dla RX/RZ.
- Align ustawia oś Z wybranego narzędzia równoległe do najbliższej osi X, Y lub Z wybranego układu współrzędnych.

TOOL, w menu JOG

- Należy sprawdzić parametry ruchu krokowego (włączanie, wartość kroku) przed wykonaniem jakichkolwiek ruchów

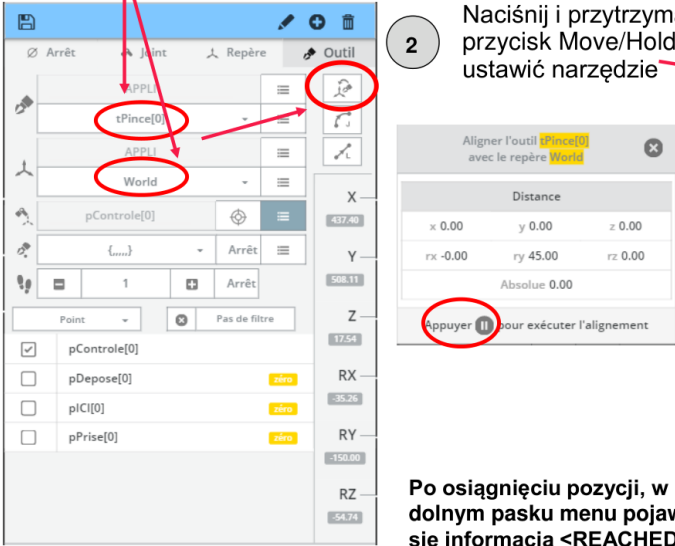


- Wprowadź wartość kroku
- Włącz tryb krokowy
- Przykład: 10 mm względem osi X, Y, Z lub 10 ° obrotu RX, RY lub RZ
- Pamiętaj o wyłączeniu trybu krokowego aby powrócić do trybu JOG

Ruch krokowy – wartość kroku i aktywacja.

wybranego narzędzia **równoległe** do najbliższej osi X, Y lub Z **aktualnie** wybranego układu współrzędnych

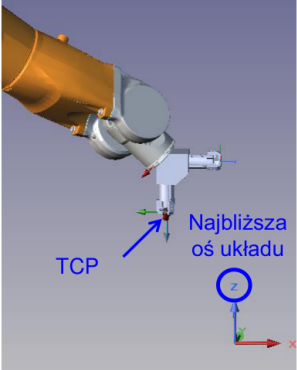
2 Naciśnij i przytrzymaj przycisk Move/Hold, aby ustawić narzędzie



Distance		
x 0.00	y 0.00	z 0.00
rx -0.00	ry 45.00	rz 0.00
Absolue 0.00		

Appuyer **[H]** pour exécuter l'alignement

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



TCP Najbliższa oś układu

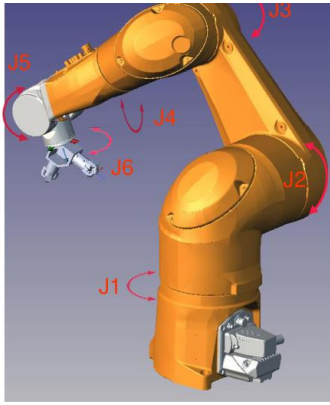
Align – ustawienie orientacji narzędzia; po osiągnięciu pozycji pojawia się REACHED

6. Nauka punktów

Ważne: Przed nauką punktu kartezjańskiego POINT zawsze wybierz właściwe narzędzie. Użycie Flange zamiast rzeczywistego narzędzia zapisze inną pozycję TCP.

Punkt JOINT

- Przechowuje położenie kątowe wszystkich osi J1–J6.
- Zalecane nazwy zaczynają się od małej litery „j”, np. jHome, jDepart.
- Wymusza konkretną konfigurację ramienia; pomaga w omijaniu kolizji i punktów osobliwych.



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

Punkt JOINT – osie i przykładowa konfiguracja ramienia.

Nauka JOINT

1. Ustaw ramię w wymaganej pozycji w trybie ręcznym i JOG.
2. Wybierz właściwe narzędzie.
3. Wybierz przeuczany punkt.
4. Naciśnij przycisk nauki punktu i zatwierdź OK.
5. Zapisz zmiany.

4 Naciśnij przycisk nauki punktu i potwierdź przyciskiem OK

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

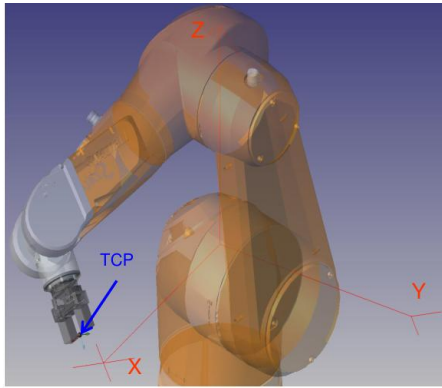
Coordinates	saved	new
j1	51.74	95.50
j2	26.56	70.07
j3	74.64	62.99
j4	0.00	0.00
j5	65.69	33.83
j6	0.00	0.00

5 Nie zapomnij zapisać zmian

Końcowe kroki nauki JOINT: teach/OK, status pozycji i zapis.

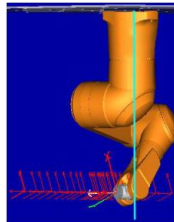
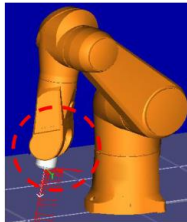
Punkt POINT

- Przechowuje X, Y, Z w mm oraz RX, RY, RZ w stopniach dla TCP w wybranym układzie.
- Zalecane nazwy zaczynają się od „p”, np. pPick, pPlace, pTraj[].
- Konfiguracja ramienia może się zmieniać; dla jednej pozycji POINT istnieje do 8 konfiguracji.
- POINT stosuje się m.in. do dokładnych pozycji pobierania/odkładania i trajektorii kartezjańskich.



Przykłady nazw : pPick, pPlace, pTraj[]

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



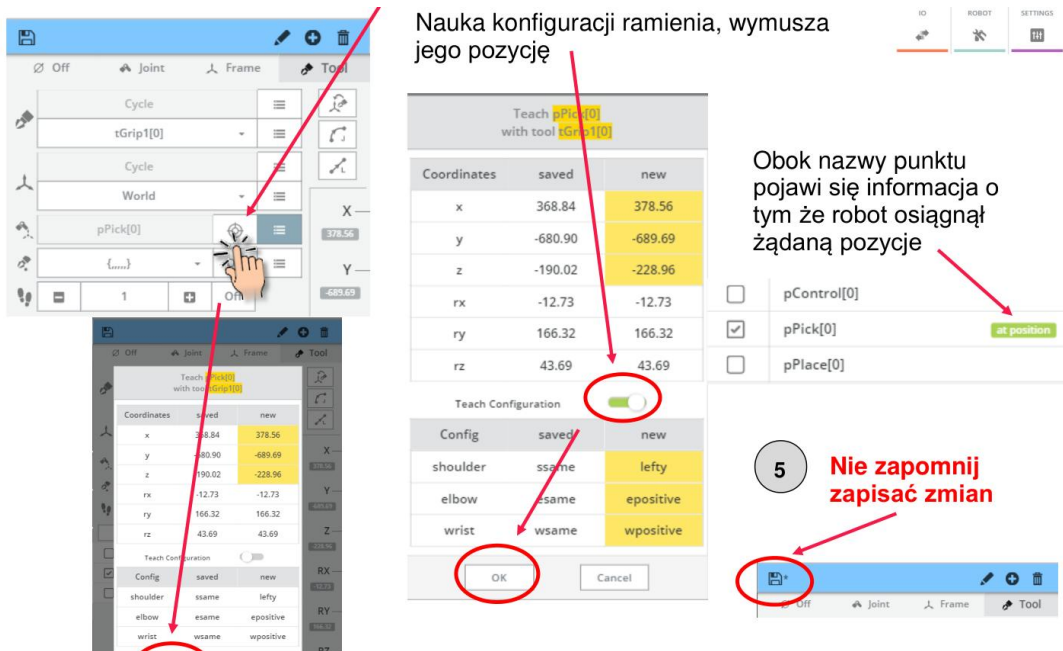
Używane do:

- 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)

Punkt POINT – pozycja i orientacja TCP w układzie współrzędnych.

Nauka POINT

1. Ustaw ramię w wymaganej pozycji.
2. Wybierz właściwe narzędzie.
3. Wybierz punkt POINT.
4. Naciśnij teach i zatwierdź OK.
5. W razie potrzeby zapisz konfigurację ramienia.
6. Zapisz aplikację/punkty.



Nauka konfiguracji ramienia, wymusza jego pozycję

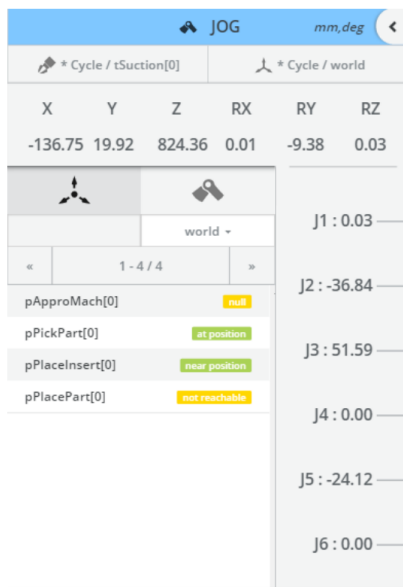
Obok nazwy punktu pojawi się informacja o tym że robot osiągnął żądaną pozycję

5 Nie zapomnij zapisać zmian

Nauka POINT – zatwierdzenie, konfiguracja ramienia, status i zapis.

Status punktu i zapis

- Nienauczony: współrzędne 0.
- Robot na pozycji: tolerancja około 0,01 mm.
- Robot w pobliżu punktu: około 1 mm.
- Pozycja nieosiągalna dla wybranego narzędzia – odpowiedni status ostrzegawczy.
- Gwiazdka obok ikony zapisu oznacza niezapisane zmiany.



- **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

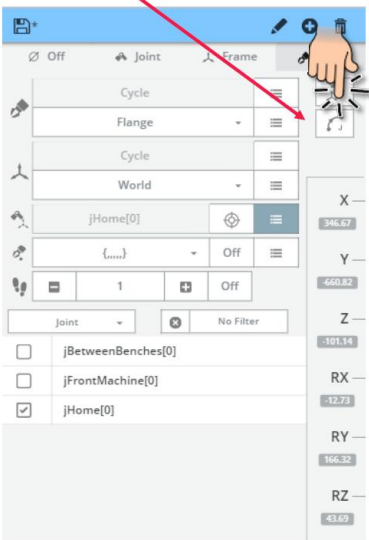
Przykładowe statusy punktu w JOG.

7. Dojazd do punktów – POINT MODE

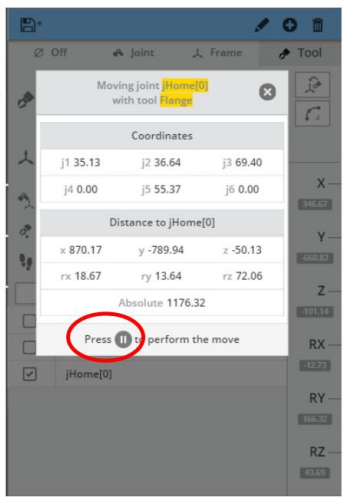
Dojazd do JOINT

- Wybierz punkt JOINT i tryb dojazdu Joint.
- Naciśnij i trzymaj Move/Hold.
- Robot jedzie po najkrótszej trajektorii osiowej – przed ruchem sprawdź możliwość kolizji.

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

Dojazd do punktu JOINT.

Dojazd do POINT

- Wybierz właściwe narzędzie.
- Do punktu POINT można dojechać ruchem JOINT (po krzywej) lub LINE (po prostej).
- Trzymaj Move/Hold do osiągnięcia punktu.

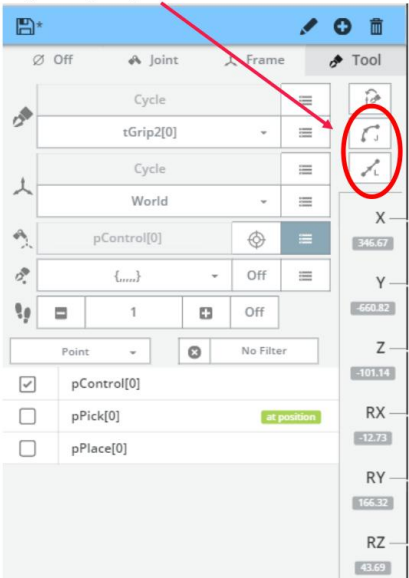
1 Przyciski  pozwalają na dojazd w trybie JOINT (po krzywej) lub LINE (po prostej) do wybranych punktów

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



Coordinates		
x	y	z
655.51	-526.04	-153.50
Distance to pControl[0]		
x	y	z
114.56	-307.19	-93.79
rx	ry	rz
0.00	0.00	0.00
Absolute 341.01		

Press  to perform the move

Dojazd do punktu POINT – wybór ruchu i narzędzia.

Dojazd nad punkt POINT (approach)

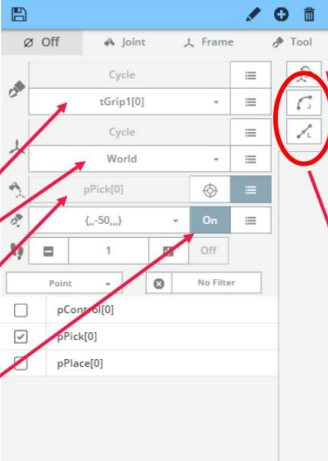
- Dostępny tylko dla POINT.
- Można ustawić odległość nad punktem oraz aktywować/dezaktywować tę funkcję.
- Na ekranie widać narzędzie, układ, punkt, offset i wybrany typ ruchu.

3 Aktywacja/deaktywacja 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
- Aktywację dojazdu nad punkt



Appro		
x	y	z
0.00	0.00	-50.00
rx	ry	rz
0.00	0.00	0.00
Coordinates		
x	y	z
346.67	-660.82	-101.14
rx	ry	rz
-12.73	166.32	43.69
Distance to pPick[0]		
x	y	z
325.34	327.79	-1168.56
rx	ry	rz
-0.00	-161.40	42.16
Absolute 1256.51		

Press  to perform the move

Parametry dojazdu nad punkt POINT.

8. Instrukcje ruchu VAL3

movej – interpolacja Joint

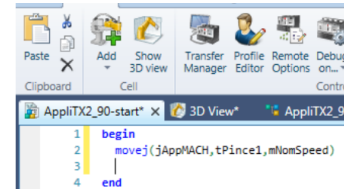
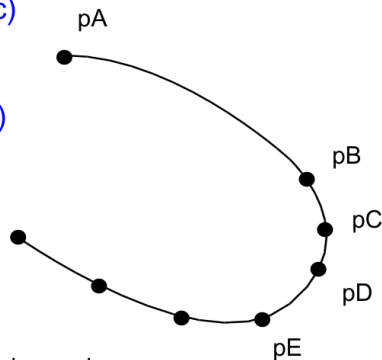
- Składnia: `movej(point, tool, mdesc)` lub `movej(joint, tool, mdesc)`.
- Ruch po krzywej w przestrzeni osi; szybki i bez punktów osobliwych.
- Stosowany do przejazdów, gdy nie trzeba zachować prostoliniowej trajektorii TCP.

```
movej(pA, tGrip, mFast)
movej(pB, tGrip, mFast)
```

`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,...

movej – przejazd po trajektorii osiowej.

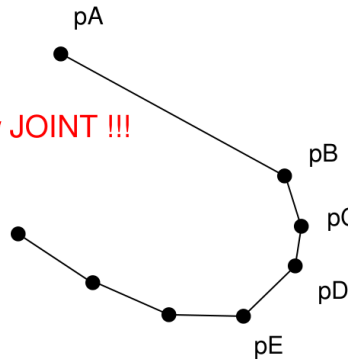
movei – interpolacja liniowa

- Składnia: `movei(point, tool, mdesc)`. Nie działa dla punktów JOINT.
- TCP porusza się po linii prostej.
- Możliwe punkty osobliwe.
- Stosowany przy pobieraniu, odkładaniu i omijaniu przeszkód wymagających kontrolowanej linii.

```
movei(pPick, tGrip, mSlow)
```

Movel(point,tool,mdesc)

Nieemożliwe dla punktów JOINT !!!



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

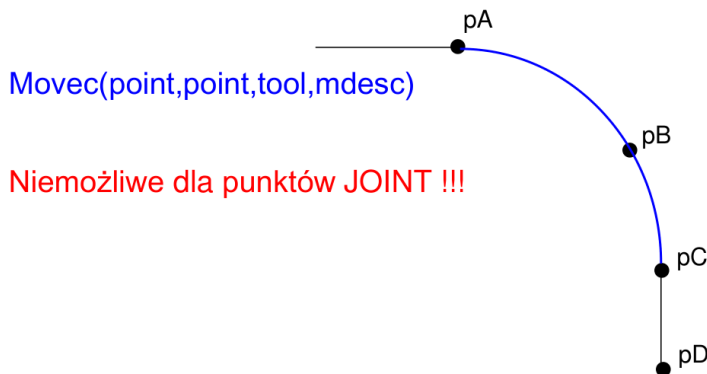
- Interpolacja Kartezjańska: ruch po linii prostej
- Prędkość i przyspieszenia zapisane w zmiennych motion descriptor (Mdesc)
- Występują punkty osobliwe => mniejsza prędkość ruchu
- Tryb wykorzystywany gdy występują: przeszkody, pobieranie, odkładanie,...

move1 – liniowy ruch TCP między punktami.

movec – ruch po łuku

- Składnia: movec(point_pośredni, point_końcowy, tool, mdesc).
- Ruch kołowy wymaga punktów POINT.
- Punkt pośredni musi tworzyć poprawną geometrię łuku; niedozwolona jest konfiguracja bez rozwiązania.

```
move1(pA, tGrip, mFast)
movec(pB, pC, tGrip, mFast)
```



Movec(point,point,tool,mdesc)

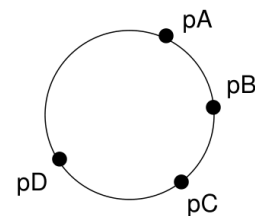
Nieemożliwe dla punktów JOINT !!!

Movel(pA,tGrip,mFast)

Movec(pB,pC,tGrip,mFast)

Movel(pD,tGrip,mFast)

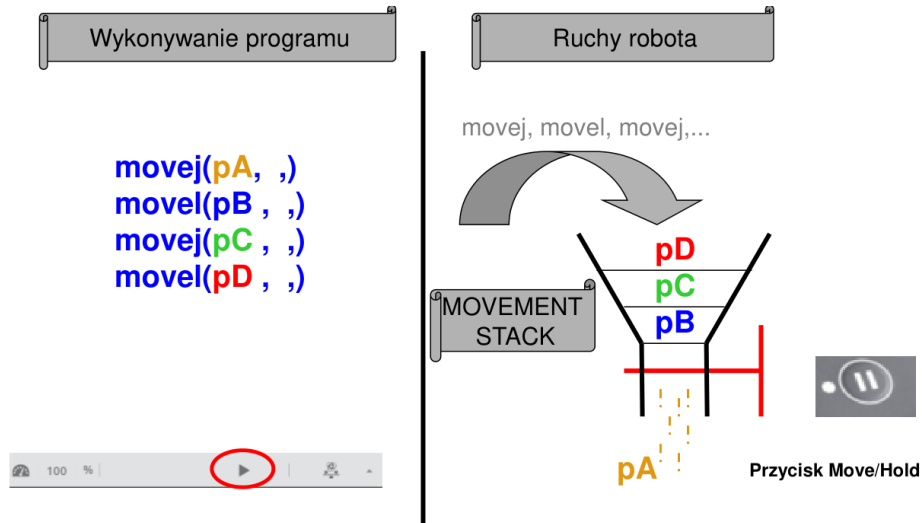
- Interpolacja kołowa: ruch po łuku
- Okrąg można zaprogramować za pomocą 4 punktów:
 - Movec(pB,pC,tGrip,mFast)
 - Movec(pD,pA,tGrip,mFast)



movec – interpolacja kołowa po łuku.

Program a fizyczny ruch robota

- Wykonywanie kodu i wykonywanie ruchu są rozdzielone przez stos ruchu (Movement Stack).
- Zmiany punktów mogą wymagać zatrzymania, przeładowania i ponownego uruchomienia aplikacji, aby zostały użyte.
- Move/Hold może wstrzymać ruch robota niezależnie od części wykonania programu.



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

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

Rozdzielenie wykonania programu i fizycznego ruchu ramienia.

waitEndMove()

- Wstrzymuje wykonywanie programu do fizycznego osiągnięcia punktu przez ramię.
- Powoduje zatrzymanie w punkcie i wyłączenie blendingu dla tego miejsca.
- Stosuj tylko tam, gdzie robot rzeczywiście ma się zatrzymać.

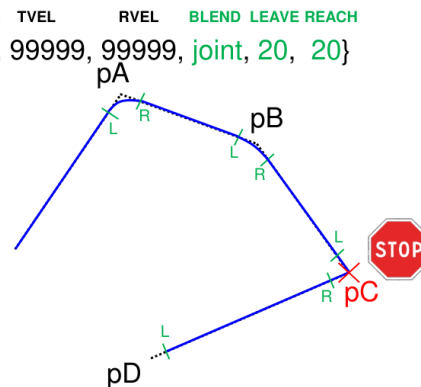
```
movej(pA, tGripper, mFast)
movej(pB, tGripper, mFast)
waitEndMove()
movej(pD, tGripper, mFast)
```

- 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)

- Przykład: mRapid = {100, 100, 100, 99999, 99999, **joint, 20, 20**}

```

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

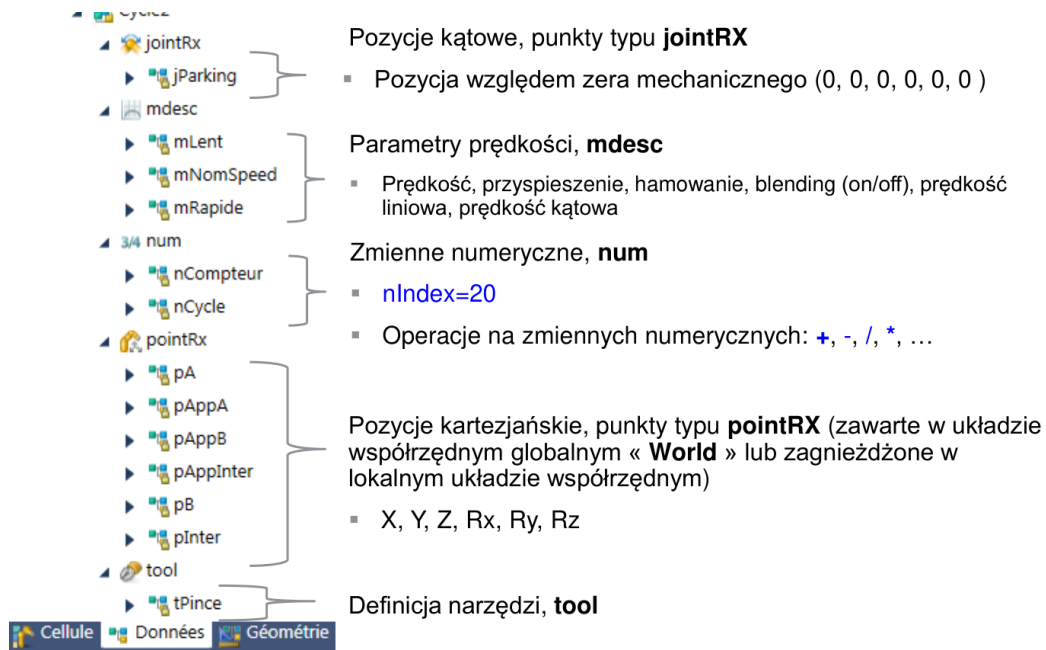


- **waitEndMove()** wykorzystujemy tylko gdy robot musi się zatrzymać w punkcie

waitEndMove() – synchronizacja programu z ruchem oraz wpływ na blending.

9. Global Data – najważniejsze typy danych

- **jointRX / joint:** pozycje osiowe J1–J6 względem zera mechanicznego.
- **pointRX / point:** pozycje kartezjańskie X, Y, Z, RX, RY, RZ w World lub lokalnym układzie współrzędnych.
- **tool:** definicja geometrii narzędzia/TCP.
- **mdesc:** parametry ruchu: prędkość, przyspieszenie, hamowanie, blending, limity liniowe i kątowe.
- **num:** zmienne liczbowe i działania arytmetyczne.
- **bool:** true/false.
- **DI/DO i analogowe I/O:** sygnały wejść/wyjść.
- Lokalny układ współrzędnych: X, Y, Z, RX, RY, RZ.
- **trsf:** transformacje kartezjańskie.
- **Tablice:** zmienne mogą być jedno- lub wielowymiarowe, np. traj[1], length[3].



Zestaw podstawowych typów Global Data.

Motion Descriptor (mdesc)

- vel, accel, decel są podawane w %.
- Tvel to maksymalna prędkość TCP w mm/s.
- Rvel to maksymalna prędkość obrotowa TCP w deg/s.
- Jeżeli nie trzeba ograniczać Tvel/Rvel, slajdy zalecają pozostawienie 99999.
- Spośród ograniczeń prędkości sterownik wykorzystuje najmniejsze.

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 spośród 3 parametrów prędkości, wykorzystany jest ten **najmniejszy**

Najważniejsze parametry Motion Descriptor.

Blending

- joint – wygładzanie ruchem osiowym; najszybsze, ale tor TCP jest mniej przewidywalny.
- cartesian – TCP pozostaje w płaszczyźnie wyznaczonej przez punkty wejścia/wyjścia i punkt blendingowany.
- off – brak wygładzania, TCP zatrzymuje się w punkcie.
- Parametry leave/reach określają obszar rozpoczęcia i zakończenia wygładzania.

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

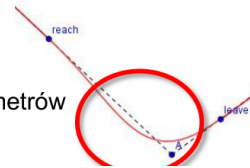
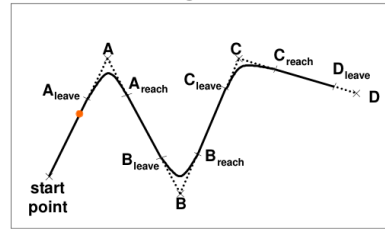
Blend: wygładzanie trajektorii:

- **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

TCP: Tool Center Point

Duże różnice wartości parametrów LEAVE i REACH

..... Basic trajectory, without blending
— with blending

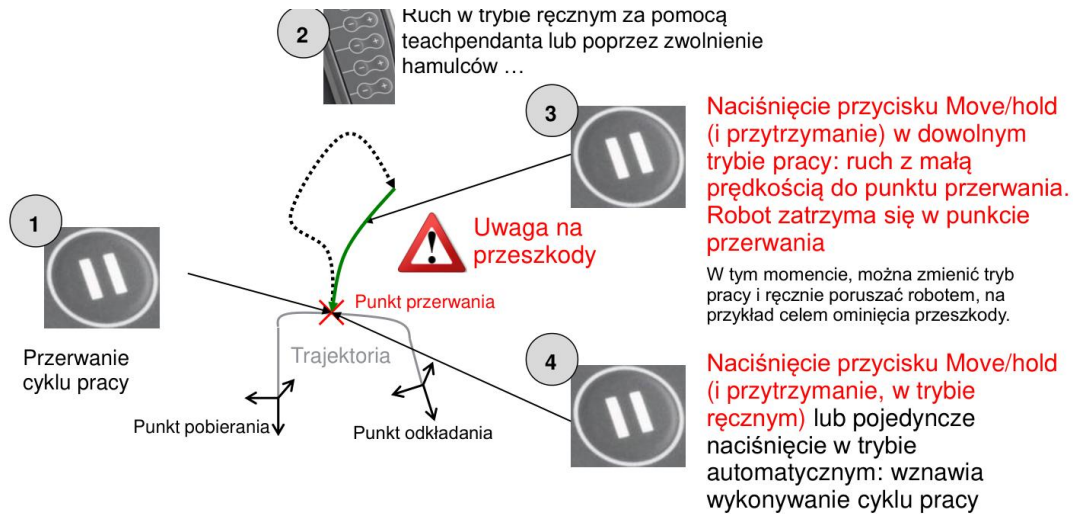


Blending: joint, cartesian i off oraz parametry leave/reach.

10. Connection Move – wznowienie po przerwaniu cyklu

- Po przerwaniu cyklu zapamiętywany jest punkt przerwania trajektorii.
- W trybie ręcznym można odsunąć ramię, np. aby ominąć przeszkodę.
- Naciśnięcie i przytrzymanie Move/Hold powoduje powolny powrót do punktu przerwania.
- Po osiągnięciu punktu przerwania można wznowić cykl: w ręcznym trzymając Move/Hold, w automatycznym pojedynczym naciśnięciem.
- Operację można wykonać bez zatrzymywania aplikacji.

Ważne: Przed powrotem do punktu przerwania sprawdź całą drogę ruchu – slajd wyraźnie ostrzega o przeszkodach.



Wszystkie czynności bez zatrzymywania aplikacji!

Connection Move – powrót do punktu przerwania i wznowienie cyklu.

11. Checklist przed uruchomieniem programu

1. Sprawdź aktywny tryb pracy i stan Safety/E-STOP.
2. Sprawdź załączenie serwonapędów i Deadman w trybie ręcznym.
3. Zweryfikuj właściwe narzędzie oraz układ współrzędnych.
4. Sprawdź, czy punkty są nauczone i zapisane.
5. Dla POINT upewnij się, że użyto właściwego TCP, a nie przypadkowo Flange.
6. Sprawdź typ przejazdu: movej / movel / movec.
7. Sprawdź mdesc: prędkości i blending.
8. Pierwszy przejazd wykonuj z małą prędkością i bądź gotowy użyć Move/Hold.
9. Jeżeli robot nie rusza, sprawdź LOGGER i statusy punktów.

Zakres źródłowy: „Szkolenie z obsługi robotów Stäubli – cz. 1, poziom operator”, slajdy 1–88. Sekcja Application Manager (slajdy 36–47) została pominięta zgodnie z poleceniem.

12. Synchronizacja ruchu z I/O i oczekiwanie

- **waitEndMove()** zatrzymuje wykonywanie programu do czasu zakończenia fizycznego ruchu robota. Używaj jej, gdy operacja I/O musi nastąpić dopiero po osiągnięciu pozycji.
- **delay(t)** wstrzymuje program na podany czas w sekundach.
- **wait(warunek)** czeka, aż warunek logiczny stanie się prawdziwy, np. sygnał czujnika.
- Przy synchronizacji ruchu i I/O materiał zaleca wyłączenie blendingu na poprzedzającym ruchu, jeżeli robot ma rzeczywiście zatrzymać się w punkcie.

```
movej(pControl, tGrip, mFast)
waitEndMove()
io:oOutput = true
delay(2.5)
io:oOutput = false
```

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

`waitEndMove()`

np.

`movej(pControl,tGrip, mFast)`

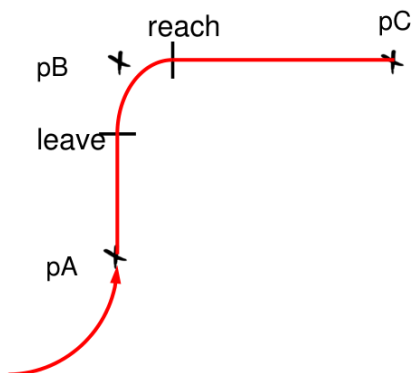
`waitEndMove()`

`DoStartcontrol=true`

....

Należy wyłączyć blending na poprzedzającym ruchu

Synchronizacja dojazdu z obsługą wejść/wyjść – waitEndMove().



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

Przykład oczekiwania na sygnał czujnika z instrukcją wait().

13. Transformacje TRSF i funkcja APPRO

- **trsf** opisuje przesunięcie i orientację za pomocą X, Y, Z, RX, RY, RZ.
- **appro(point, trsf)** oblicza punkt kartezjański przesunięty względem punktu bazowego.
- APPRO jest wygodne do tworzenia pozycji pośrednich bez uczenia każdego punktu ręcznie: najpierw dojazd nad detal, następnie ruch liniowy do punktu roboczego.
- Transformacje można składać i odwracać; w dalszej części materiału pojawiają się operacje mnożenia transformat i transformata odwrotna.

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

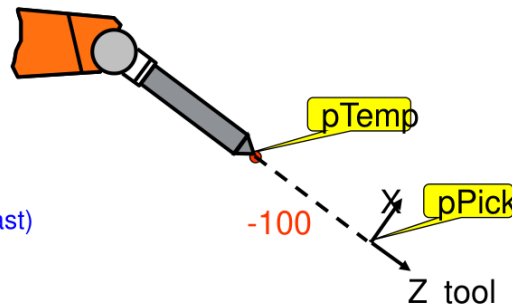
POINT <== appro(POINT,TRSF)

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)



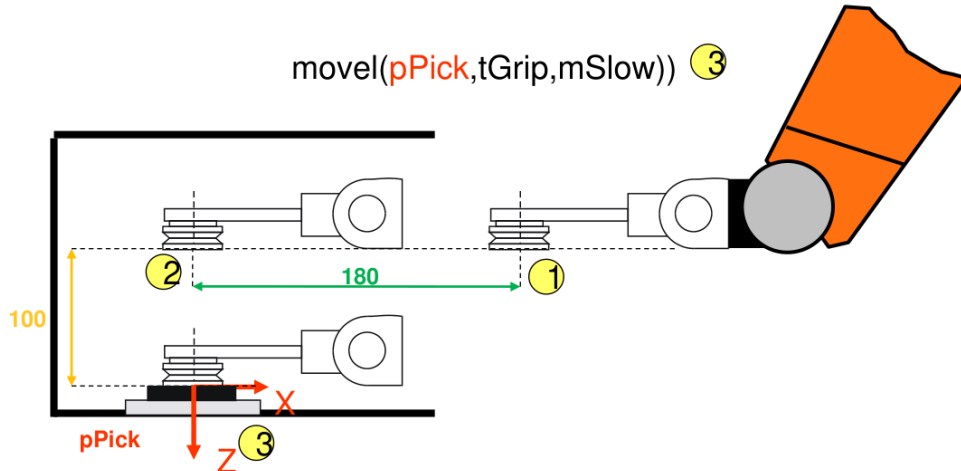
appro związane jest z układem narzędzia – dojazd do punktu

Funkcja APPRO – wyliczanie punktu przesuniętego względem pozycji bazowej.

`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



Przykład złożonego dojazdu do punktu z pozycją pośrednią.

14. Układy współrzędnych i paletyzacja

- Układy lokalne upraszczają przeuczanie wielu punktów, duplikowanie pozycji i obliczanie pozycji na paletach.
- Układ współrzędnych można nauczyć z trzech punktów: Origin, X i Y. Do nauki należy używać precyzyjnego narzędzia, a punkty referencyjne powinny być możliwie daleko od siebie. W interfejsie SP2 tryb ruchu w układzie współrzędnych jest oznaczony angielskim słowem „Frame”.

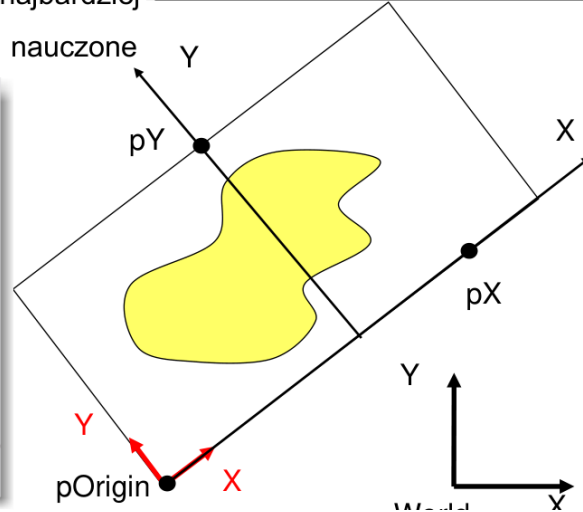
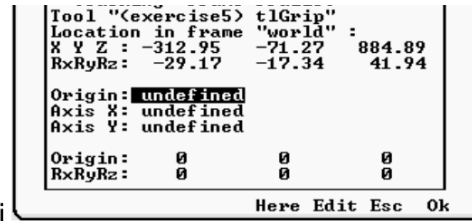
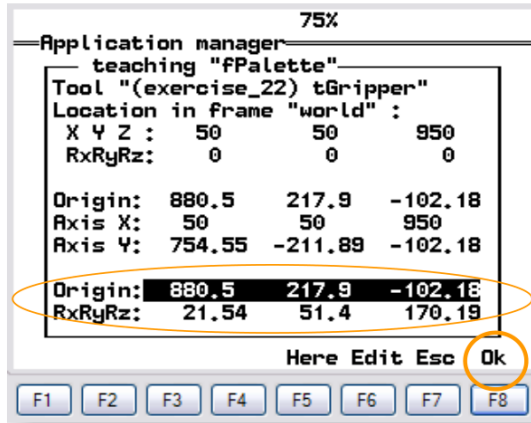
Zasada prawej dłoni – ważna przy nauce układu współrzędnych

Kierunki osi tworzą układ prawoskrętny: $\mathbf{X} \times \mathbf{Y} = \mathbf{Z}$. Ustaw palec wskazujący prawej dłoni w kierunku +X, środkowy w kierunku +Y – kciuk wskaże kierunek +Z. **Położenie trzeciego punktu określającego płaszczyznę OXY ma więc bezpośredni wpływ na kierunek osi Z.** Jeżeli punkt Y zostanie wskazany po przeciwnej stronie osi X, kierunek Y, a wraz z nim kierunek Z, zostanie odwrócony. Przed zapisaniem układu sprawdź, czy dodatni kierunek Z odpowiada zamierzonemu kierunkowi ruchu robota.

- **setFrame()** pozwala obliczyć układ współrzędnych programowo na podstawie trzech punktów.
- Punkty zapisane w drzewie danego układu współrzędnych są liczone względem tego układu; w instrukcji ruchu nie trzeba ponownie podawać układu odniesienia.
- **compose(point, frame, trsf)** służy do wyliczania kolejnych pozycji np. na paletce przez dodawanie przesunięcia.
- Przy dwóch identycznych paletach można kopiować współrzędne względne punktu między układami, zamiast ponownie uczyć wszystkie pozycje.

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

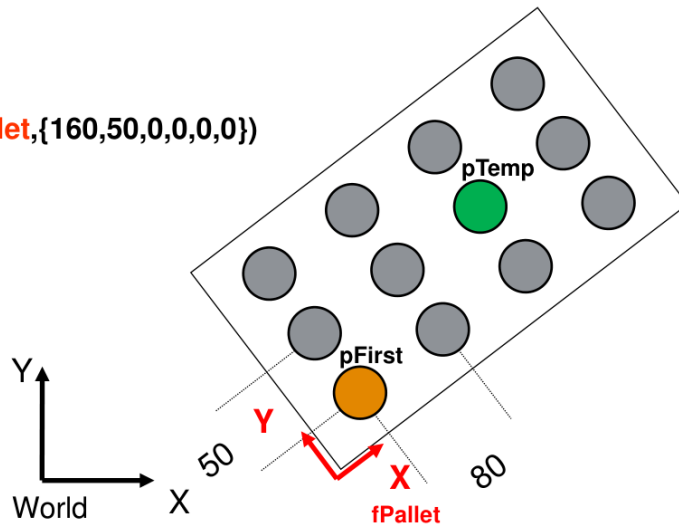


Nauka lokalnego układu współrzędnych z trzech punktów.

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})

move(pTemp,tGrip,mSlow)



Paletyzacja w lokalnym układzie współrzędnych z wykorzystaniem compose().

Dodawanie i edycja układu współrzędnych na SP2

Poniższa procedura pokazuje praktycznie, jak dodać nowy układ współrzędnych lub poprawić istniejący bezpośrednio z panelu SP2. Układ uczy się metodą 3 punktów: Origin, punkt na osi X oraz punkt wyznaczający płaszczyznę OXY. Do nauki używaj możliwie precyzyjnego narzędzia (np. pointera), a punkty wybieraj jak najdokładniej i możliwie daleko od siebie.

Krok 1. Wejdź do JOG i wybierz tryb ruchu w układzie współrzędnych.

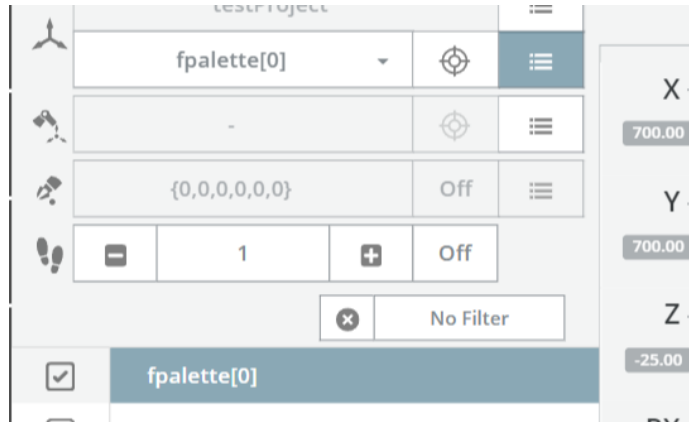
W górnej części ekranu wybierz aplikację, w której zapisany będzie układ, ustaw aktywne narzędzie oraz bazowy układ odniesienia (najczęściej world[0]). W dolnej liście widoczne są istniejące układy współrzędnych powiązane z aplikacją.



JOG → ruch w układzie współrzędnych: wybór aplikacji, narzędzia, układu odniesienia i listy dostępnych układów.

Krok 2. Dodaj nowy układ lub wybierz istniejący do edycji.

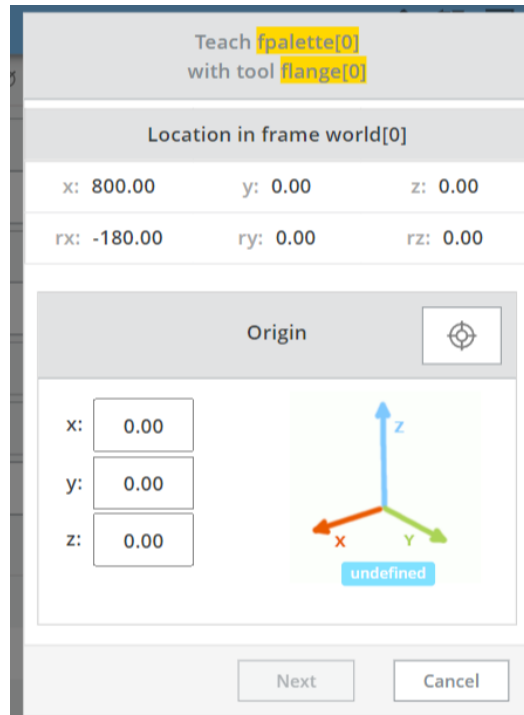
Zaznacz istniejący układ współrzędnych z listy, jeżeli chcesz go poprawić, albo użyj przycisku „+”, aby utworzyć nowy. Po zaznaczeniu układu przejdź do jego nauki/edycji.



Lista układów współrzędnych – można wskazać istniejący układ lub dodać nowy.

Krok 3. Naucz punkt Origin.

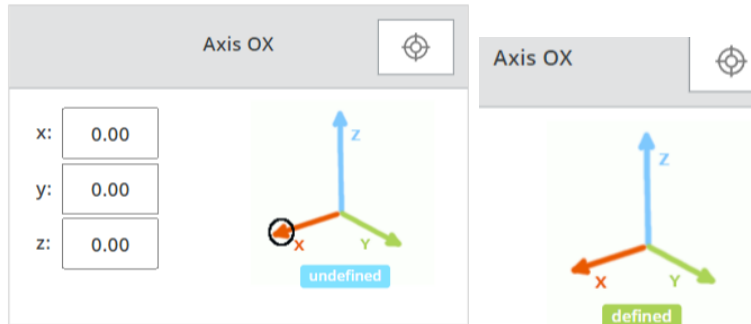
W oknie Teach panel pokazuje nazwę uczonego układu oraz narzędzie używane do nauki. Najpierw ustaw TCP dokładnie w punkcie początku układu (Origin) i zatwierdź pomiar.



Okno Teach – najpierw definiowany jest punkt Origin układu współrzędnych.

Krok 4. Naucz punkt na dodatniej osi X (Axis OX).

Przenieść TCP w punkt leżący na dodatnim kierunku osi X i zatwierdź. Przed nauką status jest niezdefiniowany, a po poprawnym zatwierdzeniu panel pokazuje już oś jako defined.



Nauka osi X – wskazanie punktu na dodatnim kierunku osi OX. Po zatwierdzeniu punktu dla osi X panel oznacza tę część definicji jako defined.

Krok 5. Naucz płaszczyznę OXY / kierunek osi Y.

Następnie wskaż trzeci punkt leżący w płaszczyźnie OXY, najlepiej w kierunku dodatniej osi Y. Na tej podstawie system wyznacza orientację całego układu współrzędnych. Pamiętaj o zasadzie prawej dłoni: wybór strony, po której znajduje się punkt Y, decyduje również o tym, w którą stronę będzie skierowana dodatnia oś Z.



Po nauczaniu trzeciego punktu płaszczyzna OXY jest zdefiniowana.

Krok 6. Sprawdź stare i nowe współrzędne, a następnie zaakceptuj zmianę.

Po zakończeniu nauki panel pokazuje porównanie zapisanych współrzędnych oraz nowych wartości. Sprawdź szczególnie x, y, z oraz orientację rx, ry, rz. Jeżeli nowe położenie jest poprawne, zatwierdź zmianę.

Coordinates	saved	new
x	100.00	800.00
y	-700.00	0.00
z	25.00	0.00
rx	0.00	0.00
ry	0.00	0.00
rz	0.00	90.00

Krok 7. Zapisz zmiany w aplikacji.

Po powrocie do widoku głównego sprawdź ikonę zapisu. Jeżeli przy ikonie dyskietki pojawia się gwiazdka, oznacza to niezapisane zmiany. Po nauczaniu lub edycji układu współrzędnych zawsze wykonaj zapis na panelu SP2.



Ikona dyskietki z gwiazdką – przypomnienie, że po nauce układu współrzędnych trzeba zapisać zmiany.

Wskazówki praktyczne

- Do nauki układu używaj precyzyjnego narzędzia i poruszaj robotem powoli, najlepiej z małą prędkością JOG.
- Punkt Origin, punkt na osi X i punkt określający płaszczyznę OXY powinny być od siebie możliwie oddalone – poprawia to dokładność definicji układu współrzędnych.
- Punkty technologiczne utworzone wewnątrz danego układu współrzędnych będą później liczone względem tego układu, co ułatwia paletyzację i przenoszenie programu na podobne stanowiska.

15. Programowanie strukturalne VAL3

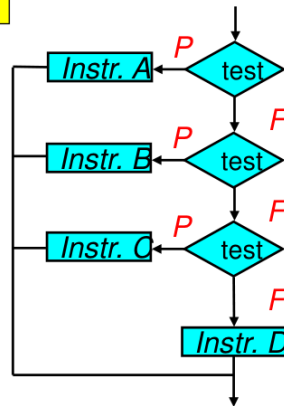
- **if / elseif / else** – wykonywanie kodu zależnie od warunków; operatory porównań obejmują m.in. <, >, ==, !=, >=, <= oraz and/or/xor/!.
- **switch** – czytelny wybór jednej z wielu gałęzi na podstawie wartości.
- **for** – pętla o znanej liczbie iteracji; można użyć własnego kroku step.
- **while / do...until** – pętle warunkowe.
- **call** – wywołuje podprogram; return może zakończyć podprogram i wrócić do programu wywołującego.

```
for nIndex=0 to 50
    move1(pTraj[nIndex], tGrip, mFast)
endFor
```

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
```

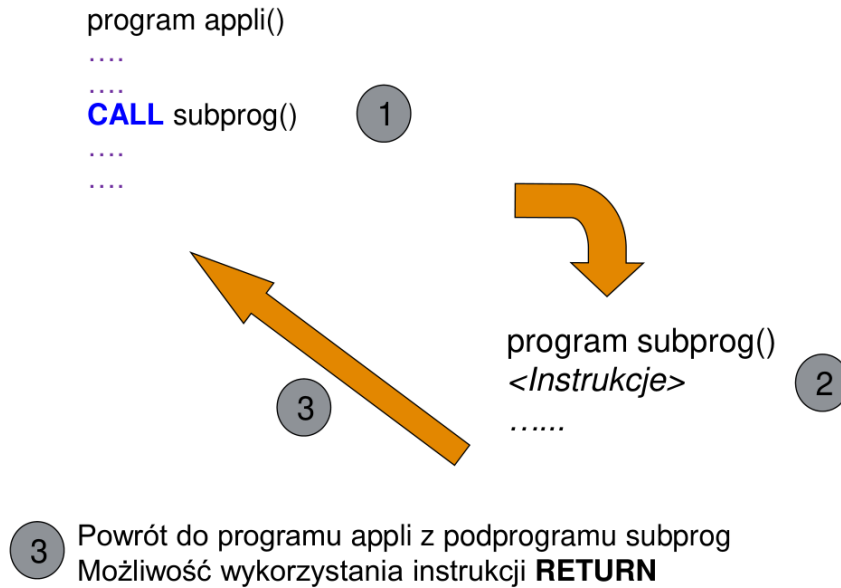
Warunki if/elseif/else i podstawowe operatory logiczne.

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

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

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

Pętla for – przykład iterowania po kolejnych pozycjach trajektorii.



Podprogramy – wywołanie call() i powrót do programu głównego.

16. Typy zmiennych VAL3 i ich zastosowanie

W VAL3 zmienne nie służą wyłącznie do przechowywania liczb. Typ określa, jakie dane są zapisane oraz jakie operacje można na nich wykonywać. W praktyce warto rozdzielić zmienne proste (num, bool, string), typy związane z ruchem robota (point, joint, tool, mdesc, config, trsf oraz lokalny układ współrzędnych) oraz zmienne komunikacyjne I/O.

16.1. Podstawowe typy proste

Typ	Co przechowuje	Typowe zastosowanie	Przykład z materiału
num	wartość liczbowa	liczniki pętli, indeksy tablic, odległości, czasy, wyniki obliczeń, kody błędów	nIndex, nTemperature, nDist, nErr
bool	true / false	flagi stanu, wyniki sprawdzeń, warunki zakończenia, informacja czy obliczenie się powiodło	bEnd, bOk
string	ciąg znaków	komendy tekstowe, komunikaty HMI, logowanie i porównywanie tekstu w if/switch	sCommand, sText
Di / Do	cyfrowe wejścia / wyjścia	wymiana sygnałów binarnych z urządzeniami zewnętrznymi	sygnały czujników, sterowanie wyjściem
Analog I/O	wartości analogowe	wejścia i wyjścia analogowe robota	wartości procesowe / sygnały analogowe
Serial	dane komunikacyjne	wymiana danych przez RS232/422 lub TCP/IP socket	komunikacja z urządzeniami zewnętrznymi

num – zmienne numeryczne

- Służą do obliczeń arytmetycznych (+, -, *, /), liczników i indeksów. W materiale num jest używany m.in. jako nIndex w pętłach for, nTemperature w warunkach oraz nDist jako wynik funkcji distance().
- Przydatne również do przechowywania czasów i wartości zwracanych przez funkcje, np. nLocalValue = clock().

```
num nIndex
num nDist
for nIndex = 0 to 50
    move1(pTraj[nIndex], tGrip, mFast)
endFor
```

bool – zmienne logiczne

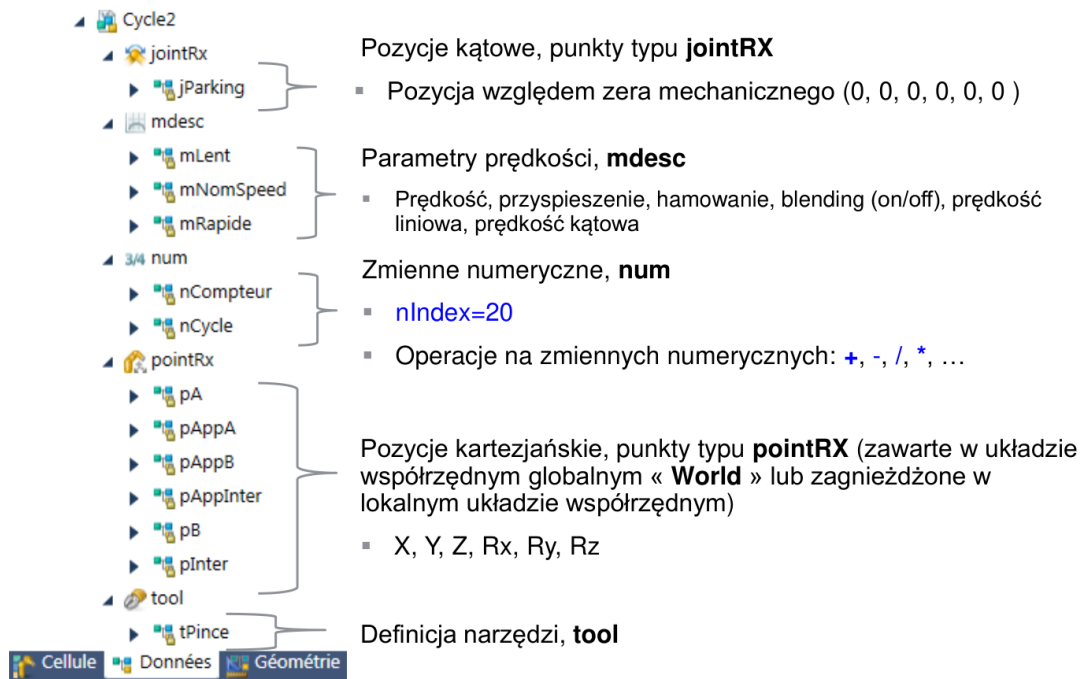
- Przyjmują wartości true lub false. Najczęściej pełnią rolę flag oraz wyniku testu wykonywanego przez funkcję.
- Przykład z materiału: bOk może przechowywać wynik sprawdzenia osiągalności pozycji lub konwersji POINT → JOINT; false oznacza brak poprawnego rozwiązania.

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

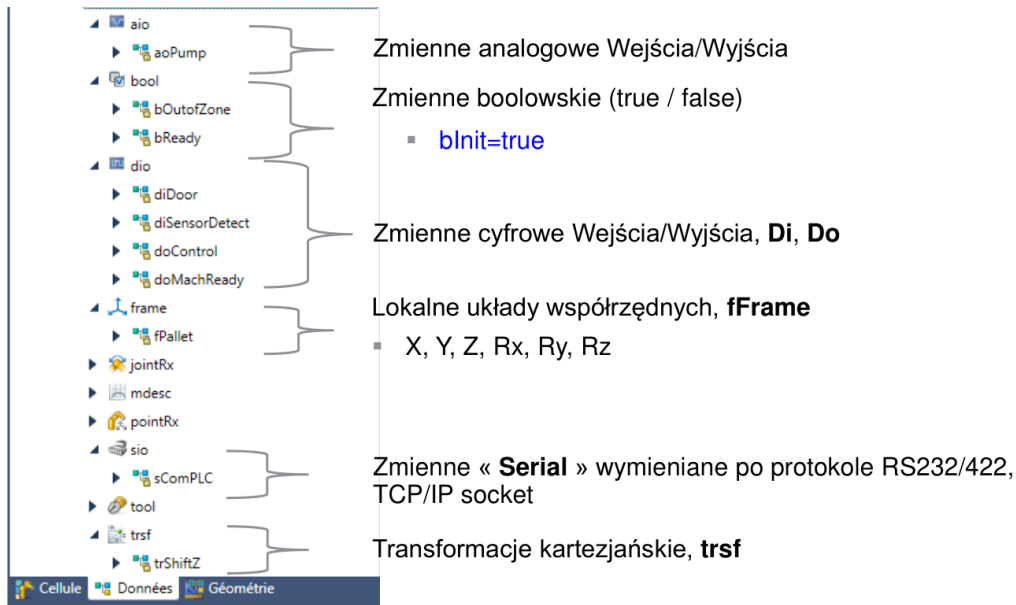
string – tekst

- Przechowuje ciągi znaków. W prezentacji string jest wykorzystywany do komunikatów i poleceń tekstowych, które można porównywać w if lub switch.

```
string sCommand
if sCommand == "Radiator On"
    ...
endIf
```



Zestaw typów Global Data: point, joint, mdesc, num i tool (slajd 76).



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

- Np.: `traj[1]`, `length[3]`, ...

Dalsze typy Global Data: lokalne układy współrzędnych, I/O, Serial, bool, trsf i tablice (slajd 77).

16.2. Typy związane z pozycją i ruchem robota

point – pozycja kartezjańska TCP

- Zawiera transformację .trsf oraz konfigurację .config. Punkt jest powiązany z układem współrzędnych, w którym został zdefiniowany.
- Zastosowanie: dokładne pozycje pobierania/odkładania, trajektorie kartezjańskie, pozycje w paletyzacji i ruchy move/movec.
- Kopiowanie pA = pB kopiuje punkt razem z odniesieniem do układu współrzędnych. Kopiowanie pA.trsf = pB.trsf kopiuje same współrzędne transformacji bez zmiany układu odniesienia punktu pA.
- Można zmieniać pojedyncze składowe, np. `pA.trsf.x = pA.trsf.x + 100`.
- Funkcje z materiału: `here(tGrip,fRef)` odczytuje bieżącą pozycję; `distance(pt1,pt2)` oblicza odległość; `position(pt,fRef)` zwraca transformację trsf punktu w danym układzie współrzędnych.

```
point pA, pB
pA = pB
pA.trsf = pB.trsf
pA.trsf.x = pA.trsf.x + 100
pCurrent = here(tGrip, fRef)
nDist = distance(pA, pB)
```

joint – pozycja osiowa J1...J6

- Przechowuje położenia katowe osi robota. Można przypisywać cały zestaw wartości albo poszczególne składowe j1...j6.
- Zastosowanie: pozycje bazowe/startowe, punkty pośrednie wymuszające konfigurację ramienia oraz bezpieczne przejazdy w przestrzeni osi.

- Obsługiwane są porównania i działania +, -, >, <. Funkcja herej() zwraca aktualną pozycję osiową, isInRange(jStart) sprawdza osiągalność, a config(jPosition) zwraca konfigurację.

```
joint jStart
jStart = {0,10,20,30,40,50}
jPosition = herej()
bOk = isInRange(jStart)
```

lokalny układ współrzędnych

- Przechowuje transformację .trsf lokalnego układu względem World. World jest układem bazowym i nie może być zmieniany.
- Zastosowanie: stoły, palety, przyrządy i obiekty technologiczne. Zmiana położenia układu współrzędnych pozwala zachować logikę punktów zdefiniowanych względem tego układu.
- Na zmiennej układu współrzędnych można wykonywać podobne operacje jak na transformacji punktu, np. przesuwać fTable.trsf.x.

```
frame fTable
fTable.trsf.x = fTable.trsf.x + 250
```

tool – narzędzie i TCP

- Opisuje geometrię narzędzia przez .trsf. Materiał pokazuje również pola .gripper, .otime i .ctime.
- Zastosowanie: prawidłowe obliczenie TCP podczas nauki i ruchu, obsługa chwytaka oraz czasów otwierania/zamykania.
- Narzędzie jest przekazywane do movej/movel/movec razem z punktem i mdesc.

```
tool tGrip
tGrip.gripper = true
tGrip.otime = 0.2
tGrip.ctime = tGrip.otime
```

mdesc – parametry ruchu

- Zawiera m.in. .vel, .accel, .decel, .tvel, .rvel, .blending, .leave i .reach.
- vel/accel/decel są wartościami procentowymi; tvel ogranicza prędkość liniową TCP w mm/s, a rvel prędkość kątową w deg/s.
- blending określa sposób wygładzania trajektorii, a leave/reach odległości rozpoczęcia i zakończenia wygładzania.
- Materiał podkreśla, że spośród ograniczeń prędkości aktywne jest to, które w danej sytuacji jest najmniejsze.

```
mdesc mFast
mFast.vel = mFast.vel / 2
mFast.blending = joint
mFast.reach = mFast.leave = 20
```

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)	
=	
==	
!=	

Pola zmiennej mdesc i przykładowa modyfikacja parametrów ruchu (slajd 190).

config – konfiguracja ramienia

- Opisuje trzy elementy konfiguracji: .shoulder, .elbow i .wrist.
- Może zostać odczytana z point/joint, skopiowana między punktami albo wymuszona przez zmianę wybranego pola, np. wrist.
- Zastosowanie: kontrolowanie postawy robota przy dojeździe do punktu kartezjańskiego i unikanie niepożądanego zmiany konfiguracji.

```
config conf
conf = pA.config
pB.config = pA.config
pA.config.wrist = wpositive
```

trsf – transformacja kartezjańska

- Reprezentuje przesunięcie i orientację (X, Y, Z, Rx, Ry, Rz). Jest składową point, lokalnego układu współrzędnych i tool.
- Transformacje można składać operatorem * i odwracać operatorem !. Umożliwia to tworzenie offsetów, przeliczanie układów i obliczanie geometrii narzędzia.

```
tr3 = tr1 * tr2
tr2 = !tr1 * tr3
tTool.trsf = !pTemp.trsf * pRef.trsf
```

16.3. Tablice, kolekcje i zmienne definiowane przez użytkownika

- Wszystkie zmienne mogą występować jako tablice jedno- lub wielowymiarowe. Materiał podaje przykłady traj[1] i length[3].
- Tablice są szczególnie wygodne dla trajektorii i paletyzacji, ponieważ można iterować po kolejnych elementach indeksem nIndex.
- Przy tworzeniu parametru programu materiał rozróżnia Element (tablica rozmiaru 1), Array i Collection.
- VAL3 pozwala także budować typy definiowane przez użytkownika. Pola takiej zmiennej mogą grupować dane należące do jednego obiektu lub działania, np. wyjście, czas i tekst.

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

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)
```

Variable

Fields

3

Zmienne definiowane przez użytkownika – grupowanie pól w jednym typie (slajd 217).

17. Zmienne globalne, lokalne i przekazywanie parametrów

17.1. Zasięg zmiennych

- Zmienne globalne są wspólne dla wszystkich programów aplikacji. Ułatwiają współdzielenie danych, ale materiał zwraca uwagę na ryzyko konfliktu, szczególnie gdy kilka programów korzysta z tych samych danych.
- Zmienne lokalne istnieją tylko w obrębie konkretnego programu. Są niezależne od innych programów, dlatego dane potrzebne w podprogramie trzeba przekazać jako parametry.
- Zmienne lokalne muszą być zadeklarowane w programie. Mogą powstać jako kopia istniejącej zmiennej albo wynik obliczenia/funkcji.

```
pLocal = pPick
nLocalValue = clock()
pLocal = here(tGrip, world)
```

Zmienne globalne:



Wspólne dla wszystkich programów

Ryzyko wystąpienia konfliktu!

Zmienne lokalne:

Niezależne od innych programów

konieczne przekazywanie do innych programów

Porównanie zasięgu zmiennych globalnych i lokalnych (slajd 210).

17.2. Parametry podprogramów

- Liczba, kolejność i typ parametrów wywołania muszą odpowiadać deklaracji podprogramu.
- Przekazanie przez wartość tworzy lokalną kopię. Zmiana parametru w podprogramie nie zmienia oryginalnej zmiennej w programie głównym.
- Przekazanie przez referencję zapisuje parametr z &, np. num& x_nResult. Podprogram pracuje wtedy na tej samej zmiennej, więc zmiana jest widoczna po powrocie do programu głównego.

```
program multiplication(num x_nValue1, num x_nValue2, num& x_nResult)
begin
    x_nResult = 0
    ...
end
```

- W praktyce przekazanie przez wartość jest odpowiednie dla danych wejściowych, których podprogram nie powinien zmieniać, a referencja dla wyniku lub danych, które mają zostać zmodyfikowane przez podprogram.

18. Polecenia systemowe i diagnostyka programu

- **clock()** – czas systemowy w sekundach; przydatny do pomiaru czasu cyklu.
- **isPowered()** – informacja, czy serwonapędy są załączone.
- **isCalibrated()** – informacja o stanie kalibracji.
- **isSettled()** – informacja, czy robot zakończył ruch / zatrzymał się.
- **isEmpty()** – sprawdza, czy stos poleceń ruchu jest pusty.
- **workingMode()** – zwraca aktualny tryb pracy i może podać dodatkowe informacje o stanie.
- **getProfile()/setProfile()** – odczyt i zmiana profilu użytkownika z poziomu VAL3.
- **esStatus()** – odczyt stanu układu bezpieczeństwa.
- **enablePower()/disablePower()** i funkcje prędkości monitora dotyczą sterowania napędami oraz ustawieniami pracy; należy stosować je zgodnie z ograniczeniami trybu pracy opisanymi w instrukcji.

isPowered(): zwraca TRUE jeżeli są załączone serwonapędy;

isCalibrated(): zwraca TRUE 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;

Wybrane instrukcje systemowe do kontroli stanu robota i stosu ruchu.

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;

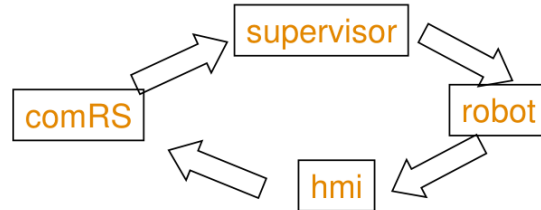
workingMode(), sterowanie napędami i prędkością w odpowiednich trybach pracy.

19. Wielozadaniowość

- Wielozadaniowość pozwala uruchamiać kilka niezależnych programów jednocześnie, np. ruch robota, nadzór celi i komunikację.
- **taskCreate** uruchamia zadanie asynchroniczne z nazwą i priorytetem.
- Priorytet 1–100 określa, ile linii zadania może zostać wykonanych przed przekazaniem czasu innym zadaniom.
- **wait()**, **delay()**, **waitEndMove()** i część operacji systemowych powodują przejście wykonania do innego zadania.
- **taskKill** / **taskSuspend** / **taskResume** / **taskStatus** służą do zarządzania cyklem życia zadań.

- Program START jest uruchamiany po RUN, a STOP po zatrzymaniu lub zakończeniu aplikacji; po STOP zadania są usuwane zgodnie z opisem w materiale.
- taskCreateSync() służy do zadań synchronicznych wykonywanych w określonym cyklu czasowym.

- **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.

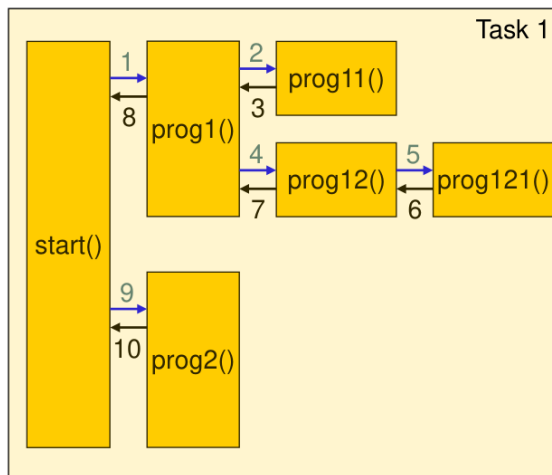


▪ Przykład :

- 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»

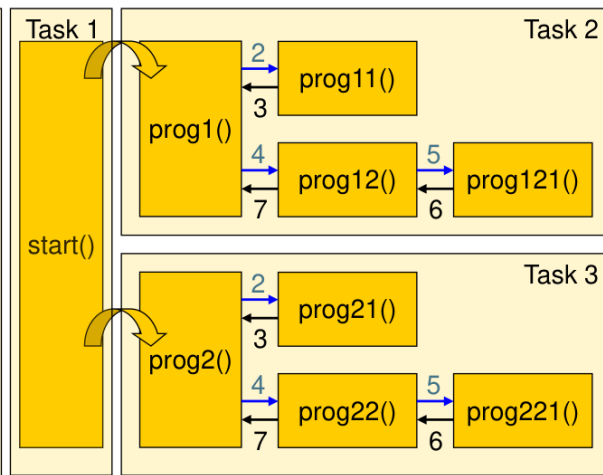
Idea wielozadaniowości – niezależne programy dla ruchu, nadzoru i innych funkcji.

Przebieg programów **call()**:



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

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



Wykonywanie równoległe,
Wiele potoków wykonywania programów

Różnica między sekwencyjnym call() a równoległym taskCreate().

`taskKill("secu")` zakańcza zadanie

`taskSuspend("secu")` wstrzymuje wykonywanie zadania

`taskResume("secu",jump)` wznowia 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

Podstawowe polecenia zarządzania zadaniami.

20. Biblioteki

- Biblioteki służą do ponownego użycia danych i programów w wielu aplikacjach.
- Elementy, które mają być widoczne z zewnątrz, muszą zostać oznaczone jako publiczne.
- Aplikacja deklaruje interfejs biblioteki i nadaje mu lokalny identyfikator.
- Dostęp do elementów biblioteki odbywa się przez kwalifikowaną nazwę, np. `libRef1:pA` lub `call libRef1:init()`.
- `libLoad()/libSave()` umożliwiają ładowanie i zapisywanie bibliotek; instrukcje zwracają kody błędów, które należy kontrolować.

```
nErr = lib:libLoad("ref1")
movej(lib:pA, tGrip, mFast)
lib:nX = 10
nErr = lib:libSave()
```

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

Cel stosowania bibliotek – wspólne dane i programy dla wielu aplikacji.

- 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()`

Przykłady użycia punktu, zmiennej i programu z biblioteki.

21. Calibration i Adjustment

- **Calibration** przywraca zgodność pozycji kątowych osi zapisanych w sterowniku z rzeczywistym położeniem mechanicznym robota.
- **Adjustment** jest procedurą dokładniejszego ustawienia zależności pomiędzy pozycją mechaniczną i odczytem układu pomiarowego.
- Materiał opisuje także sposób wyznaczania pozycji na podstawie liczby obrotów i pozycji enkodera; procedury te są częścią obsługi serwisowej, a nie typowej codziennej pracy operatora.

Adjustment

- **Calibration** (**F5 Rec** - *recovery*) – kalibracja polega na przywróceniu położenia kąowego 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

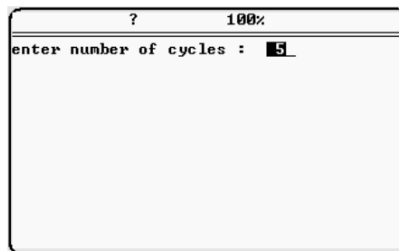
Calibration i Adjustment – rozróżnienie procedur serwisowych.

22. HMI / strony użytkownika

- Slajdy 255–263 opisują tekstowy interfejs USER stosowany w rodzinie CS8C: `userPage()`, `cls()`, `title()`, `gotoxy()`, `put()/putln()` oraz `get()`.
- `clock()` i `getDate()` mogą być używane do prezentowania czasu i daty na stronie użytkownika.
- Dla CS9 materiał pokazuje tworzenie graficznych stron użytkownika powiązanych z aplikacją VAL3.
- Widżety są rozmieszczane graficznie, a ich wartości mogą być wiązane z VAL3; stronę można wyświetlać na SP2 lub w przeglądarce.

Wyświetlany po naciśnięciu **USER**

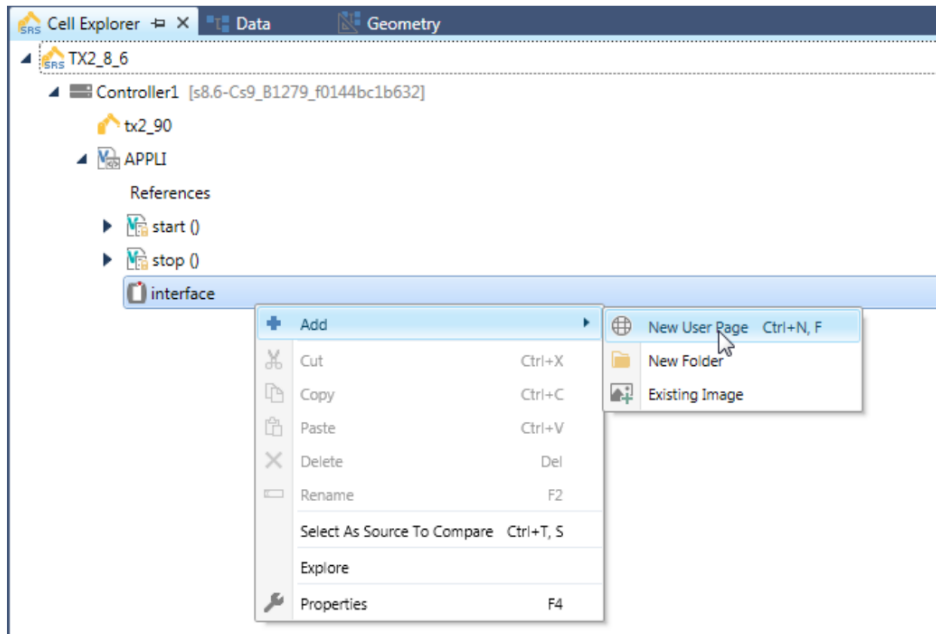
Ekran użytkownika pozwala na wyświetlanie komunikatów i wprowadzanie danych.



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

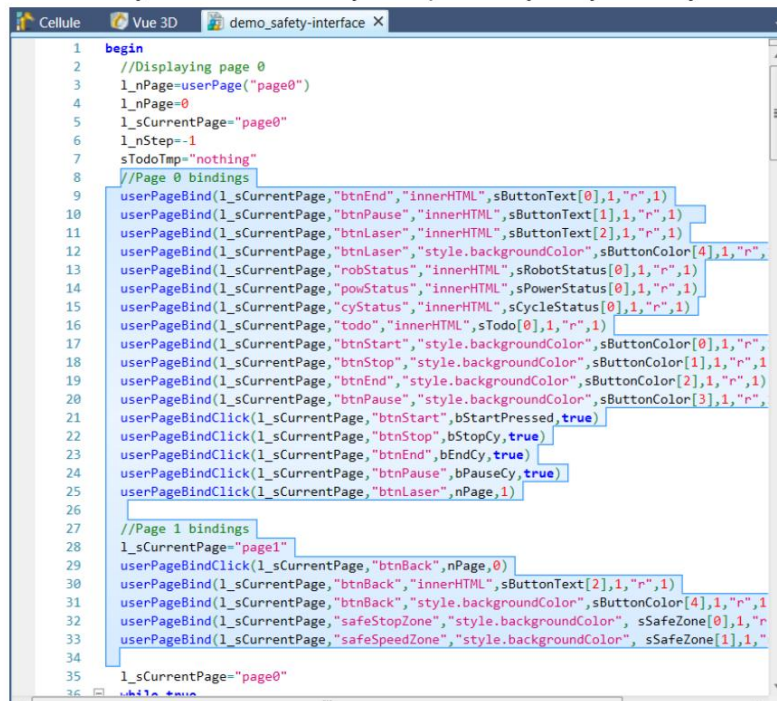
Tekstowa strona użytkownika – podstawowe funkcje wyświetlania i czyszczenia ekranu.

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



CS9 – tworzenie stron użytkownika z poziomu aplikacji VAL3.

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



Powiązanie elementów HMI ze zmiennymi i logiką VAL3.