

Systemy programowania robotów przemysłowych - obsługa i programowanie robotów Stäubli

Sterownik w wersji CS8/CS8C



1

Klasy robotów Stäubli

Małe obciążenia 1-10kg

Roboty 4-osiowe typu
SCARA TS20, TS40,
TS60, TS80

Roboty 6-osiowe serii
TX40, TX60,
TX2-40, TX2-60



Średnie obciążenia 10-80kg

Roboty 6-osiowe serii
TX90, TX2-90,
RX160



Duże obciążenia powyżej 80kg

Roboty 6-osiowe serii
TX200
TX340SH



Roboty specjalne

Przeznaczenie:

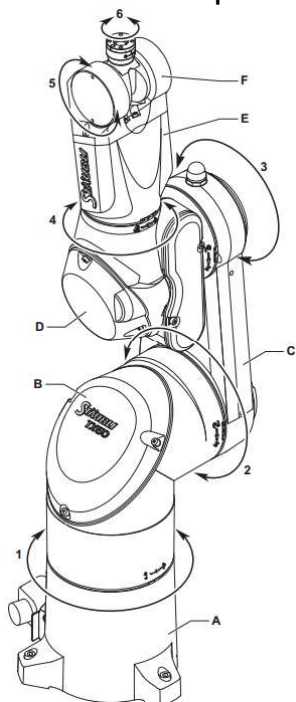
- malowanie
- przetwórstwo tworzyw sztucznych
- praca w warunkach czystych lub sterylnych
- praca w warunkach dużej wilgotności



2

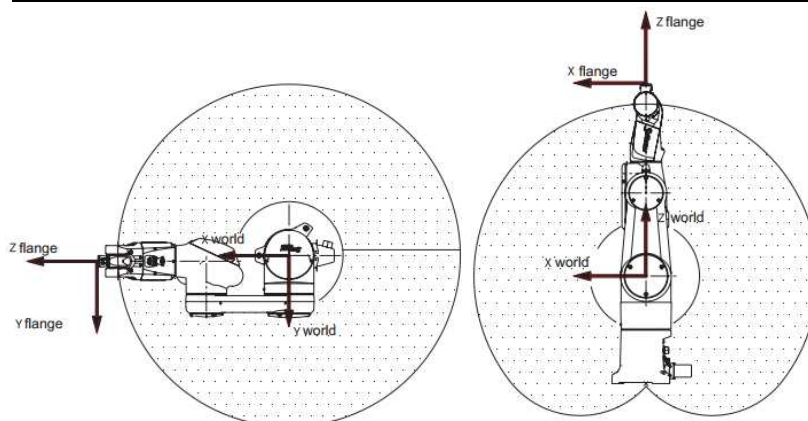
Robot Stäubli TX60, TX60L

Podstawowe parametry ramienia manipulacyjnego



podstawa (A),
bark (B), ramię (C), łokieć (D),
przedramię (E), nadgarstek (F)

	TX60	TX60L
Masa własna [kg]	51,4	52,5
Udźwig nominalny [kg]	3,5	2
Udźwig maksymalny [kg]	4,5	3,7
Maksymalny zasięg między osią 1 a 5 [mm]	600	850
Prędkość maksymalna dla środka ciężkości ładunku [m/s]	8	10,6
Prędkość ruchu ręcznego dla punktu TCP [mm/s]	250	250
Powtarzalność [mm]	±0,02	±0,03



3

Sterownik robota CS8C

Sterownik robota jest systemem mikroprocesorowym odpowiedzialnym przede wszystkim za obliczenia i sterowanie ramieniem manipulatora poprzez wzmacniacze mocy poszczególnych osi ramienia manipulacyjnego.

Główne elementy składowe:

1. Wzmacniacze mocy
2. Moduł zasilania robota (RPS – Robot Power Supply)
3. Pomocniczy moduł zasilania układów logicznych (ARPS – Auxiliary RPS)
4. Moduł układów zabezpieczeń (RSI – Robot Safety Interface)
5. Jednostka centralna systemu
6. Główny wyłącznik sterownika
7. Moduł zasilacza głównego (PSM – Power Supply Module)



4

Panel operatora i wyboru trybu pracy

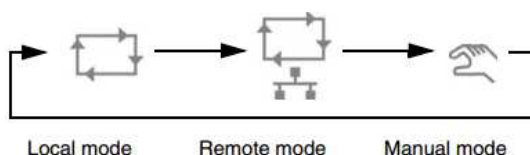
Panel operatora MCP (Manual Control Pendant)



(#) – przycisk zezwolenia na załączenie napędów (3-pozycyjny)



Panel przedni wyboru trybu pracy WMS (Working Mode Selection) - umożliwia za pomocą stacyjki wybór jednego z trzech trybów pracy. Wybrany tryb pracy sygnalizowany jest przez kontrolki na MCP i WMS.



5

Panel operatora MCP

Panel operatora MCP - opis funkcji przycisków

- (1) Przycisk wyboru trybu pracy z sygnalizacją wybranego trybu (gdy w systemie jest panel WMS przycisk jest nieaktywny a wybór następuje przez WMS)
- (2) Przycisk załączenia zasilania napędów robota
- (3) Stop awaryjny
- (4) Przyciski sterowania ręcznego w wybranym układzie
- (5) Przyciski wyboru trybu sterowania ręcznego












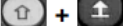


- (6) Przycisk ustawiania prędkości ruchu dla danego trybu pracy
- (7) Przyciski funkcyjne do wyboru opcji z menu wyświetlanego powyżej na ekranie panelu
- (8) Klawisze klawiatury alfanumerycznej (numerycznej) do wprowadzania danych w aplikacjach
- (9) Przyciski obsługi interfejsu użytkownika oraz nawigacji w menu
- (10) Przyciski kontroli pracy aplikacji
- (11) Przycisk zezwolenia na załączenie napędów (#)
- (12) Przyciski aktywacji wybranych wyjść cyfrowych sterownika, 1 – zamykanie/otwieranie chwytaka
- (13) Przyciski umożliwiające poruszanie ramieniem z użyciem jednej ręki

6

Panel operatora MCP

Przyciski obsługi interfejsu użytkownika oraz nawigacji (9)







-  Przycisk do aktywacji pomocy kontekstowej (wyjście poprzez ponowne wciśnięcie przycisku Help lub Esc)
-  Przycisk wywołania głównego menu ekranowego
-  Przycisk do wywołania okna użytkownika programowania VAL3
- Zestawy przycisków wspomagających poruszanie się w oknie użytkownika VAL3
 -  Przełączenie do poprzedniego okna VAL3
 -  Przełączenie do następnego okna VAL3
 -  Przełączenie do pierwszego okna VAL3
 -  Przełączenie do ostatniego okna VAL3
-  Przycisk umożliwia przeskok do elementu listy na ekranie dla którego wprowadzono pierwszą lub pierwsze litery nazwy z klawiatury alfanumerycznej (ang. pick list)
-  Rozwinięcie elementu menu poprzedzonego znakiem „+”
-  Zwinięcie elementu menu poprzedzonego znakiem „-”
-  Przycisk zablokowania funkcji Shift, umożliwia dostęp do alternatywnych funkcji przycisków z wyjątkiem znaków „\$” i „\”
-  Dostęp do znaków „\$” i „\” oraz wielkich liter

7




Panel operatora MCP

Przyciski obsługi interfejsu użytkownika oraz nawigacji (9)

Przyciski poruszenia się po listach elementów

-  Przejście na koniec listy
-  Przejście na początek listy
-  Przewinięcie o stronę w górę
-  Przewinięcie o stronę w dół
-  Aktywacja akcji związanej z elementem, otwarcie okna do edycji parametrów, zatwierdzenie wprowadzonych wartości
-  Przełączanie między polami w otwartym oknie

Przyciski sterujące pracą aplikacji (10)

-  Przyciski stop zatrzymujące działanie uruchomionej aplikacji
-  Przycisk umożliwiający wybór i uruchomienie aplikacji
-  Przycisk aktywacji ruchu, zatrzymywania i wznowiania działania aplikacji

W trybie sterowania ręcznego ruch automatyczny w aplikacji lub dojazdu do punktu jest możliwy gdy przycisk jest wciśnięty

W trybie sterowania automatycznego i zdalnego za pomocą tego przycisku można wyzwać zatrzymanie i wznowienie ruchu robota

8

Panel operatora MCP

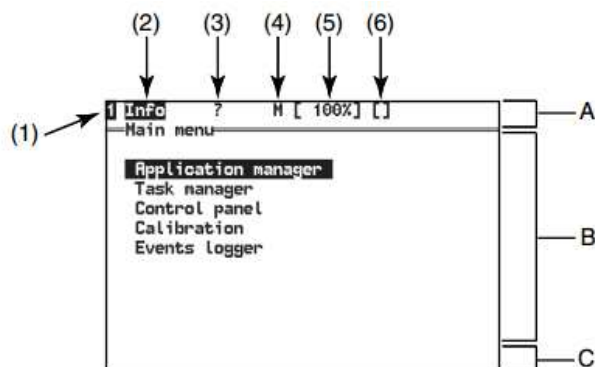
Ekran panelu MCP

Trzy sekcje ekranu MCP:

A – linia statusowa

B – główne pole pracy dla okien użytkownika

C – linia z menu kontekstowym

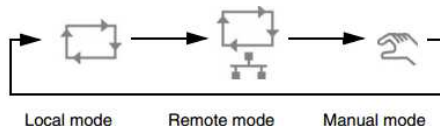


Elementy linii statusowej:

- (1) Sygnalizacja połączenia ze sterownikiem oprogramowania SRS oraz aktywności systemu. Cyfra oznacza liczbę aktywnych połączeń, w polu może się pojawiać wskaźnik zajętości systemu.
- (2) Znacznik informujący o nowych komunikatach systemowych zapisanych do logu systemowego
- (3) Znacznik określający, że aplikacja VAL3 oczekuje na wprowadzanie wpisów w oknie aplikacji
- (4) Znacznik informujący o ruchu ramienia robota, „M” oznacza wykonywanie ruchu, „S” oznacza, że ramię zostało zatrzymane w trakcie ruchu, „@” dla ruchu ręcznego oznacza osiągnięcie lokalizacji docelowej, znacznik nie jest wyświetlany, gdy kolejka wykonywanych ruchów jest pusta
- (5) Pole określające ustawioną maksymalną prędkość ruchu w danym trybie pracy
- (6) Znacznik (mrugające „[]”) informujący o zablokowaniu dostępu w aplikacji do wybranych wejść/wyjść, blokada może wprowadzona przez operatora w panelu konfiguracji systemu

9

Tryby pracy sterownika



Tryb RĘCZNY

- Przeznaczony jest do ręcznej manipulacji ramieniem oraz do przygotowania i testowania programów. Prędkość ruchu ograniczona do 250 mm/s
- Manipulacja ręczna jest wykonywana za pomocą przycisków sterowania ruchem
- Ruchy automatyczne (dojazd do punktu) mogą być wykonywane wyłącznie po wciśnięciu przycisku Move/Hold, ruch jest przerywany natychmiast po puszczeniu przycisku
- Uruchomienie programu ruchowego wymaga wczytania aplikacji z programem oraz załączenia napędów
- Jakikolwiek ruch jest możliwy przy wciśniętym przycisku zezwolenia w pozycji środkowej (po załączeniu zasilania napędów)
- W trybie ręcznym nie ma możliwości zainicjowania ruchu przez urządzenia zewnętrzne

Tryb LOKALNY

- Umożliwia wykonywanie ruchów automatycznych bez udziału operatora zgodnie z zapisanym programem z maksymalnymi prędkościami określonymi w aplikacji
- Automatyczne wykonanie programu wymaga wczytania aplikacji z programem oraz załączenia napędów (przycisk Run)
- Uruchomienie programu aplikacji następuje po wciśnięciu przycisku Move/Hold (ponowne użycie przycisku powoduje wstrzymanie/wznawianie działania)
- Operator może modyfikować maksymalną prędkość za pomocą przycisków +/-
- Zatrzymanie działania aplikacji przycisk Stop

10

Tryby pracy sterownika

Tryb ZDALNY

Tryb zdalny jest funkcjonalnie podobny do trybu lokalnego i daje możliwość zdalnego uruchomienia aplikacji. Różnice są następujące:

- Zasilanie napędów ramienia jest załączane przez system zewnętrzny (PLC, zewnętrzny MCP) poprzez cyfrowe wejście sterownika lub z użyciem polecenia **enablePower** języka VAL
- Wyzwolenie wykonywania ruchu Move/Hold może być wydawane automatycznie przy załączonym zasilaniu napędów ramienia (polecenie **autoConnectMove**)
- Przycisk Move/Hold może być nieaktywny (w zależności od uprawnień użytkownika)
- Przycisk wyłączania zasilania może być nieaktywny (w zależności od uprawnień użytkownika)
- Sygnał systemowy **enablePower** musi mieć przypisane wejście cyfrowe, załączenie zasilania następuje po podaniu na przypisane wejście impulsu trwającego 200ms, ponowne podanie impulsu wyłącza zasilanie napędów

11

Ręczna manipulacja ramieniem

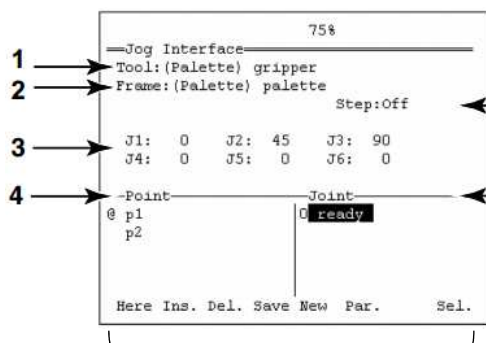
Uruchamianie zasilania napędów ramienia manipulacyjnego

- Ustawienie trybu ręcznego na WMS
- Wciśnięcie przycisku zezwolenia do pozycji środkowej (przycisk 2)
- Załączenie zasilania (przycisk 3) w ciągu 15 sekund od wciśnięcia przycisku zezwolenia

Sterowanie w trybie ręcznym uzyskuje się po wybraniu jednego z czterech możliwych trybów (grupa przycisków 4). Wybranie trybu automatycznie aktywuje wyświetlenie okna sterowania ręcznego na MCP (Jog Interface).



- 1 – wybrane narzędzie
- 2 – wybrany układ bazowy
- 3 – bieżąca pozycja zależnie od trybu pracy
- 4 – lista punktów w układzie kartezjańskim



- 6 – przemieszczenie z zadaniem krokiem (w mm lub stopniach)
- 5 – lista punktów we współrzędnych przegubowych

Przyciski funkcyjne do obsługi w oknie sterowania ręcznego

12

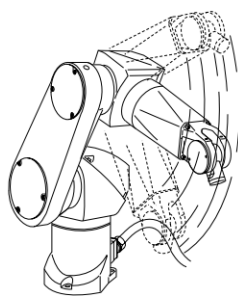
Ręczna manipulacja ramieniem

Znaczenie symboli przy nazwach lokalizacji

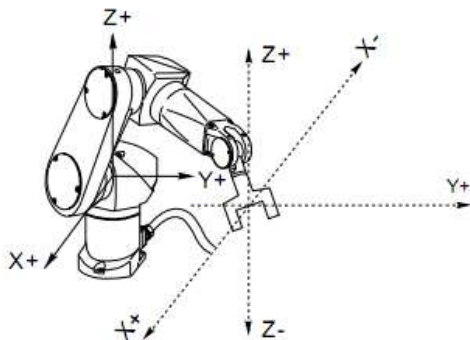
- '@' – oznacza, że końcówka robota jest dokładnie w zdefiniowanej lokalizacji
- '~' – oznacza, że robot jest blisko lokalizacji
- '0' – oznacza lokalizację z zerowymi wartościami
- '!' – oznacza punkt niemożliwy do osiągnięcia z wybranym narzędziem



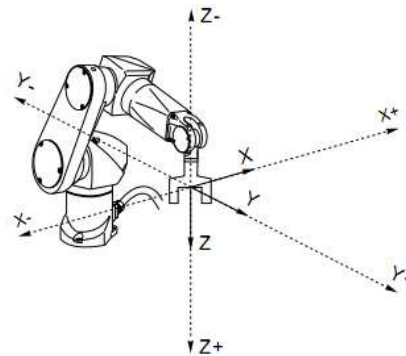
Ręczna manipulacja we współrzędnych przegubowych (JOINT) oraz we współrzędnych kartezyjskich (FRAME, TOOL)



JOINT



FRAME



TOOL

13

Ręczna manipulacja ramieniem

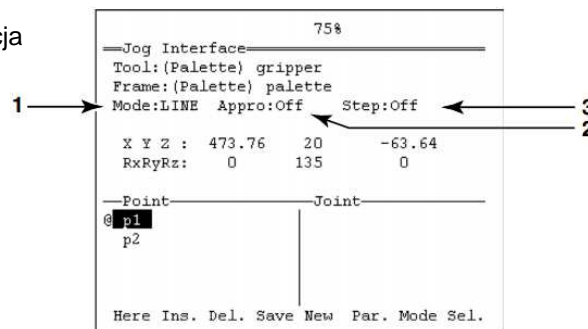
Ruch w trybie dojazdu do punktu (POINT)

- Ruch do punktu jest możliwy gdy jest otwarta aplikacja oraz zostały w niej nauczone lub zdefiniowane lokalizacje
- Możliwe są trzy tryby ruchu do punktu (aktywny tryb widoczny w miejscu 1 okna sterowania), wybór trybu jest wykonywany przez sekwencyjne wciskanie przycisku Mode z menu kontekstowego:

1 – LINE dojazd liniowy

2 – JOINT ruch w trybie od punktu do punktu

3 – ALIGN w którym oś Z narzędzia jest ustawiana równoległe do najbliższej osi układu FRAME



- Sposób ruchu do punktu można dodatkowo modyfikować aktywując opcję Appro 2 oraz krok dojazdu Step 3. Otwarcie okna do parametryzacji przycisk Par. Możliwy jest dojazd do punktu (opcja Appro aktywna) wzdłuż osi X,Y,Z lub wykonując rotacje wokół tych osi RX, RY, RZ
- Wykonanie ruchu do punktu jest realizowane za pomocą przycisku Move/Hold

14

Uruchamianie aplikacji

Uruchomienie aplikacji wymaga jej wczytania z pamięci dyskowej do pamięci operacyjnej systemu

Uruchamianie w trybie ręcznym:

- Wybór trybu ruchu ręcznego na WMS
- Wciśnięcie przycisku zezwolenia do pozycji środkowej
- Załączenie zasilania napędów (aplikacje, w których nie ma realizacji ruchu manipulatora nie wymagają załączenia napędów)
- Uruchomienie aplikacji poprzez przycisk Run
- Aktywacja ruchu poprzez przycisk Move/Hold

Uruchamianie w trybie lokalnym:

- Wybór trybu lokalnego na WMS
- Załączenie zasilania napędów
- Uruchomienie aplikacji poprzez przycisk Run
- Aktywacja ruchu poprzez przycisk Move/Hold

Uruchamianie w trybie zdalnym:

- Wybór trybu zdalnego na WMS
- Uruchomienie aplikacji poprzez przycisk Run
- Załączenie zasilania napędów następuje poprzez impuls z zewnętrznego sterownika lub z aplikacji poprzez polecenie **enablePower**.

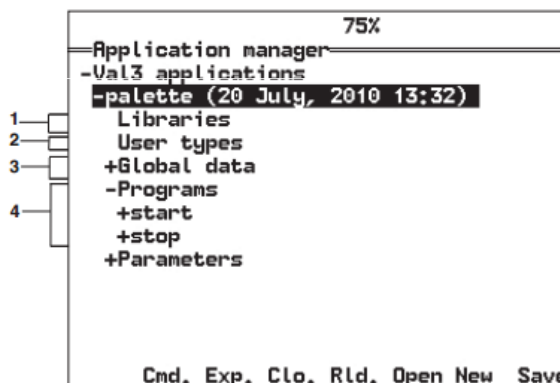
15

Menadżer aplikacji

Menadżer aplikacji (dostępny z MCP w menu głównym - Application manager) umożliwia przygotowanie, wczytywanie oraz zarządzanie aplikacjami w systemie VAL 3. Przygotowanie (wczytanie) aplikacji jest niezbędne do wykonywania ruchów w trybie dojazdu do punktu oraz programowych ruchów automatycznych.

Utworzenie nowej aplikacji w menadżerze umożliwia wybór **New** z menu przycisków kontekstowych. Domyślna, nowa aplikacji składa się z:

- (1) Libraries - bibliotek, w których można umieszczać zbiory funkcjonalności (podprogramy, funkcje) definiowane przez użytkownika,
- (2) User types - listy typów definiowanych przez użytkownika,
- (3) Global variables - zmiennych globalnych, w których można wyróżnić między innymi zmienne związane z układami i lokalizacjami Frame, Point (grupa zmiennych World) oraz narzędziami typu Tool (grupa zmiennych Flange),
- (4) Programs - dwóch programów domyślnych **start()** oraz **stop()**, które zawsze uruchamiane są odpowiednio przy uruchamianiu i kończeniu pracy aplikacji (programy te nie mogą zawierać parametrów wejściowych, nie można ich usunąć ani zmienić nazwy),
- (5) Parameters - zestawu domyślnie ustawionych parametrów aplikacji.



16

Menadżer aplikacji

Menu kontekstowe umożliwia: **Cmd. Exp. Clo. Rld. Open New Save**

- **Cmd.** – wprowadzanie i wykonywanie poleceń języka VAL 3 z linii komend (np. przez „?” można podejrzeć zawartość zmiennych)
- **Exp.** – eksport aplikacji pod nową nazwą oraz w innej lokalizacji zapisu
- **Clo.** – zamknięcie wybranej aplikacji z listy aplikacji otwartych
- **Rld.** – ponowne wczytanie aplikacji bez konieczności zamykania i otwierania
- **Open** – otwarcie aplikacji
- **New** – utworzenie nowej aplikacji
- **Save** – zapisanie na dysku aplikacji
- **Del** – usunięcie aplikacji, dostępne w menadżerze na stronie otwierania aplikacji

Tryby otwierania aplikacji w trakcie uruchamiania systemu sterownika robota (przycisk **Mode**):

- **Manual** – żadna akcja nie jest wykonywana, aplikacja wymaga ręcznego załadowania
- **Autoload** – aplikacja jest ładowana automatycznie podczas startu systemu
- **Autostart** – aplikacja jest automatycznie ładowana i uruchamiana podczas startu systemu (tylko jedna aplikacja może mieć status Autostart)

17

Menadżer aplikacji, układy narzędzia i bazowe

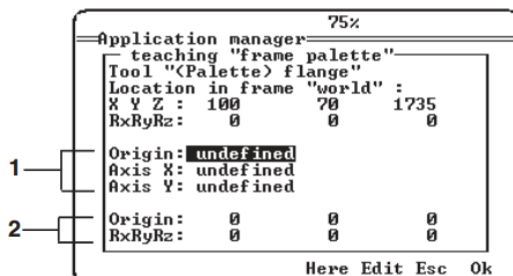
Definiowanie układów narzędzi

Definiowanie nowego narzędzia w aplikacji możliwe jest jedynie przez wprowadzenie danych numerycznych wybierając **Global data->flange** a następnie wprowadzając nową zmienną typu **Tool** (przycisk **New**) Narzędzie referencyjne typu **tool** o nazwie **flange** jest zawsze zdefiniowane w aplikacji VAL3. Jest to zmienna, która reprezentuje opis kołnierza robota (koniec łańcucha kinematycznego) służący do podłączenia każdego zdefiniowanego później narzędzia. Każde narzędzie wykorzystywane w programowaniu robota jest dołączone pośrednio lub bezpośrednio do kołnierza.

Definiowanie układów bazowych

Układy bazowe mogą być definiowane poprzez dane numeryczne oraz metodą uczenia dla znanego w systemie narzędzia.

- Dane numeryczne wprowadza się po utworzeniu nowej zmiennej globalnej typu **Frame** w grupie **world (Global data->world)**
- Nauczenie układu bazowego dostępne jest poprzez przycisk **Teac** menadżera aplikacji, który powoduje otwarcie okna uczenia. Uczenie odbywa się metodą 3-punktową poprzez wskazanie początku układu oraz punktów na osiach X i Y oraz potwierdzenie przyciskiem **Here**



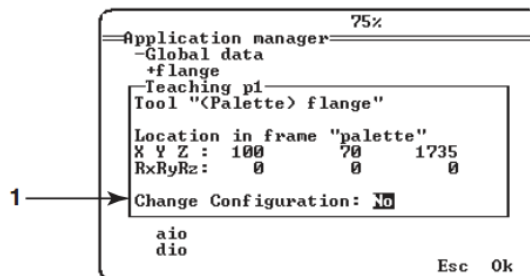
18

Menadżer aplikacji, uczenie punktów

Lokalizacje mogą być wprowadzane poprzez dane numeryczne lub metodą uczenia przez wskazywanie w menadżerze aplikacji oraz w oknie sterowania ręcznego

Menadżer aplikacji:

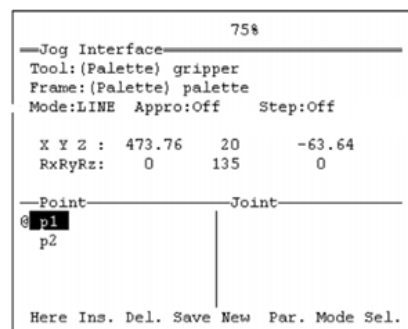
- Utworzenie nowej zmiennej typu **Point** w grupie **world** lub **Joint** w grupie **joint** lub nowego pola w odpowiedniej tablicy zmiennych już istniejących
- Wprowadzenie danych numerycznych lub ustawienie żądanej konfiguracji ramienia i zapamiętanie przyciskiem **Here**



Okno sterowania ręcznego:

- Wstawienie zmiennej typu **Point** lub **Joint** lub nowego pola w tablicy już istniejącej
- Wprowadzenie danych numerycznych lub ustawienie żądanej konfiguracji ramienia i zapamiętanie przyciskiem **Here**

W trakcie uczenia lokalizacji typu **Point** możliwe jest zadanie specyficznej konfiguracji ramienia lub pozostawienie aktualnej wynikającej z bieżącej konfiguracji ramienia



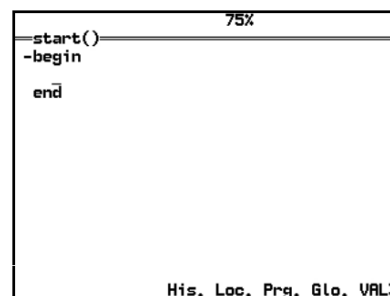
19

Edytor programów w menadżerze aplikacji

Edytor programów VAL 3 umożliwia wstawianie, usuwanie oraz modyfikację zawartości programów w aktywnej aplikacji

Główne menu kontekstowe edytora VAL 3:

- **His.** – otwiera okno 20 wprowadzonych wcześniej poleceń
- **Loc.** – wyszukanie lub utworzenie zmiennej lokalnej lub parametru
- **Prg.** – wybór lub utworzenie nowego programu
- **Glo.** – wybór lub utworzenie nowej zmiennej globalnej
- **VAL3** – otwiera okno z listą poleceń języka VAL 3



Zastosowanie w edytorze drzewiastej struktury programu

```
-if nNb>12
+switch nNb
else
  putln("Error: too many items")
endIf
```

```
-if nNb>12
  -switch nNb
    -case 5
      break
    +case 7
    +default
  endSwitch
else
  putln("Error: too many items")
endIf
```

20

Debugowanie aplikacji

Debugowanie aplikacji jest możliwe z poziomu MCP w menadżerze aplikacji (przycisk **Dbg.**) oraz z użyciem zewnętrznego oprogramowania SRS. Po wyborze opcji debugowania działająca aplikacja jest zatrzymywana na bieżąco realizowanym poleceniu.

Znaczniki widoczne w debugerze:

- * – oznacza linie programu (2) w których znajdują się punkty zatrzymania wykonywania programu (Breakpoints)
- > – wskaźnik programu który oznacza kolejną linię do wykonania (1)

```
start<> 75%
- begin
  move j<ready, flange, mNomSpeed>
  - while true
    * move j<p1, tool1, mNomSpeed>
      move j<ready, flange, mNomSpeed>
      move j<p2, tool1, mNomSpeed>
      > waitEndMove<>
    endwhile
  end
Bpts ; Data ->* <>* <*> Rsm. Save
```

Znaczenie opcji w menu kontekstowym:

- **Bpts** – wstawianie punktów zatrzymania wykonywania programu
- ; – możliwość wprowadzenia komentarza do wybranej linii
- **Data** – dostęp do danych zmiennych dla wybranego polecenia
- ->* – przesunięcie wskaźnika programu do wybranej linii bez wykonywania poleceń
- {}* – praca krokowa zgodnie z widocznymi liniami programu
- {*} – praca krokowa z wykonywaniem pojedynczych poleceń VAL 3
- **Rsm./Sus.** – wznowienie/zatrzymanie działania aplikacji bez wyjścia z debugera
- **Save** – zapisanie aplikacji na dysku
- Wyjście z debugera przycisk **Esc**

Elementy programowania języka VAL 3

Proste typy danych

- **bool** - typ logiczny (true/false),
- **num** - typ wartości numerycznych (całkowitych lub zmiennoprzecinkowych),
- **string** - typ dla łańcuchów znaków (znaki ASCII/Unicode),
- **dio** - dla wejść/wyjść cyfrowych,
- **aio** - dla wejść/wyjść analogowych,
- **sio** - dla portów wejść/wyjść, łączy szeregowych lub internetowych,
- **screen** - do wyświetlania komunikatów ekranowych MCP i dostępu do klawiatury.

Typy strukturalne

- **trsf** - typ dla transformacji geometrycznych w układzie kartezjańskim XYZ, zawiera następujące pola: x, y, z, rx, ry, rz, typ ten definiuje zmianę położenia i/lub orientacji. Jest kompozycją translacji i rotacji. Transformacja sama w sobie nie reprezentuje pozycji w przestrzeni ale jest interpretowana jako położenie i orientacja punktu lub ramki w układzie kartezjańskim względem innej ramki, np. ramki związanej z układem podstawowym,
- **frame** - typ dla reprezentacji pozycji (położenia i orientacji) w kartezjańskim układzie odniesienia (wykorzystuje pola struktury typu **trsf**), zawiera następujące pola: x, y, z, rx, ry, rz,
- **tool** - typ opisujący narzędzie, zawiera następujące pola: x, y, z, rx, ry, rz, IOlink, otime, ctime, typ ten określa geometrię narzędzia oraz sterowanie nim. Pola x, y, z służą do opisu pozycji TCP (Tool Center Point) w układzie współrzędnych robota, rx, ry, rz służą do opisu orientacji narzędzia. Pole IOlink – wyjście używane do aktywacji narzędzia, otime – czas potrzebny do otwarcia narzędzia (w sekundach), ctime – czas potrzebny do zamknięcia narzędzia.

Elementy programowania języka VAL 3

Typy strukturalne cd.

- **config** - typ dla konfiguracji manipulatora, która określa rodzaje ułożenia ogniw manipulatora tworzących bark (ogniwa: kolumna - ramię), łokieć (ogniwa: ramię - przedramię) i nadgarstek (ogniwa: przedramię - kiść), zawiera następujące pola: shoulder, elbow, wrist. Każde pole może przyjąć wartość jednego z 4 parametrów, umożliwiając określenie ułożenia ogniw w wyróżnionych przegubach:
 - shoulder (bark): righty - z prawej, lefty - z lewej, ssame - jak poprzednio, free - dowolne,
 - elbow (łokieć): epositive - do góry, enegative - do dołu, esame - jak poprzednio, efree - dowolnie,
 - wrist (nadgarstek): wpositive - do góry, wnegative - do dołu, wsame - jak poprzednio, wfree - dowolnie.
- **point** - typ reprezentujący pozycję narzędzia czyli jego położenie i orientację w układzie kartezjańskim względem układu odniesienia (pola struktury typu **trsf**) oraz konfigurację ramienia robota w tym punkcie (pola struktury typu **config**), zawiera następujące pola: x, y, z, rx, ry, rz, shoulder, elbow, wrist.
- **joint** - typ dla pozycji robota w układzie współrzędnych wewnętrznych, zawiera następujące pola: j1, j2, j3, j4, j5, j6.
- **mdesc** - typ dla parametrów ruchu robota, zawiera następujące pola: accel, vel, decel, tvel, rvel, blend, leave, reach. Typ ten przechowuje kolejno wartości: przyspieszenia robota, prędkości robota, opóźnienia robota, maksymalnej prędkości translacji TCP w mm/s, maksymalnej prędkości kątowej obrotu narzędzia wokół TCP w st./s, blend - pole wyboru rodzaju interpolacji w otoczeniu punktów pośrednich ścieżki (off/joint/Cartesian), leave - odległość w mm przed punktem pośrednim, reach - odległość w mm za punktem pośrednim.

23

Elementy programowania języka VAL 3

Wybrane polecenia związane z realizacją ruchu programowego

- **joint** herej()
Instrukcja zwraca aktualną pozycję kątową wszystkich osi robota i może być przypisana do zmiennej typu joint: **jpoint** = herej()
- **point** here(**tool** tTool, **frame** fReference)
Instrukcja zwraca aktualną pozycję w układzie kartezjańskim i może być przypisana do zmiennej typu point
- **point** appro(**point** pPosition, **trsf** trTransformation)
Instrukcja zwraca punkt zmodyfikowany przez transformację trTransformation, która jest definiowana względem tego samego układu odniesienia co argument pPosition. Wynik można podstawić do zmiennej typu **point**.
- num movej(**point** pPosition, **tool** tTool, **mdesc** mDesc),
num movej(**joint** jPosition, **tool** tTool, **mdesc** mDesc)
Instrukcja ruchu robota do punktu pPosition z wykorzystaniem narzędzia tTool oraz z parametrami ruchu mDesc. Ruch odbywa się najkrótszą drogą dla współrzędnych osiowych manipulatora.
- num movel(**point** pPosition, **tool** tTool, **mdesc** mDesc)
Instrukcja ruchu liniowego do punktu pPosition z wykorzystaniem narzędzia tTool oraz z parametrami ruchu zawartymi w zmiennej mDesc. Ruch odbywa się najkrótszą drogą dla współrzędnych kartezjańskich.
- num movec(**point** plnIntermediate, **point** pTarget **tool** tTool, **mdesc** mDesc)
Instrukcja ruchu kołowego od poprzedniej lokalizacji przez punkt plnIntermediate do punktu pPosition z wykorzystaniem narzędzia tTool oraz z parametrami ruchu zawartymi w zmiennej mDesc.

24

Elementy programowania języka VAL 3

Wybrane polecenia związane z realizacją ruchu programowego cd.

- `waitEndMove()`
Umożliwia wstrzymanie działania przedbiegu programu i realizację dokładnego dojazdu do punktu (bez omijania) z zatrzymaniem
- `open(tool tTool)`
Instrukcja otwarcia narzędzia tTool.
- `close(tool tTool)`
Instrukcja zamknięcia narzędzia tTool.

25

Elementy programowania języka VAL 3

Instrukcje kontroli przebiegu programu

- Pętla programowa odrzucająca
`while <bool bCondition>`
 blok z kodem programu
`endWhile`
- Pętla nieodrzucająca
`do`
 blok z kodem programu
`until <bool bCondition>`
Pętle ze sprawdzaniem warunku wykonują bloki z kodem programu tak długo, dopóki warunek bCondition jest prawdziwy. W pierwszym przypadku warunek jest sprawdzany przed wykonaniem bloku programu, w drugim po jego wykonaniu.
- Pętla licznikowa
`for <num nCounter> = <num nBeginning> to <num nEnd>`
 blok z kodem programu
`endFor`
Pętla pomiędzy instrukcjami `for` i `endFor` wykonuje się dopóki zmienna nCounter nie osiągnie wartości nEnd; nBeginning – wartość początkowa indeksu pętli.
- Wywołanie programu użytkownika
`call program_name([parameter1][,parameter2])`
Powrót z podprogramu możliwy jest za pomocą polecenia `return`

26

Elementy programowania języka VAL 3

Instrukcje kontroli przebiegu programu cd.

- Instrukcja warunkowa

```
if <bool bCondition>
    blok z kodem programu
elseif <bool bAlternateCondition>
    blok z kodem programu
else
    blok z kodem programu
endif
```
- Instrukcja wyboru

```
switch <expression>
case <value1> [,<value2>]
    blok z kodem programu
    break
case <value3> [,<value4>]
    blok z kodem programu
    break
default
    blok z kodem programu
    break
endSwitch
```

27

Elementy programowania języka VAL 3

Obsługa wejść/wyjść cyfrowych poprzez typ **dio**

- W aplikacji należy utworzyć zmienne globalne w grupie zmiennych **dio** oraz połączyć je z fizycznymi wejściami/wyjściami sterownika, np. modułu BasicIO (funkcja **Link**)

```
out0=false          in0=false
BasicIO-1\%Q0      BasicIO-1\%I0
bOut0              bIn0
```

- W programie należy utworzyć zmienne lokalne typu **dio** oraz połączyć za pomocą instrukcji dioLink

```
dio bout0local
dio bin0local
```

```
-begin
dioLink(bout0local,out0)
dioLink(bin0local,in0)
bout0local=true
if (bin0local==true)
    waitEndMove()
endif
end
```

- Instrukcje odczytu i zapisu grupowego wejść wyjść

num dioGet(dio diArray[]) – zwraca wartość numeryczną w postaci liczby całkowitej zapisanej w kodzie binarnym odczytaną z tablicy zdefiniowanych wejść

num dioSet(dio diArray[], num nValue) – konwertuje wartość numeryczną na liczbę całkowitą w kodzie binarnym i zapisuje do zdefiniowanej tablicy wyjść, funkcja zwraca wartość całkowitą wartości wejściowej

28