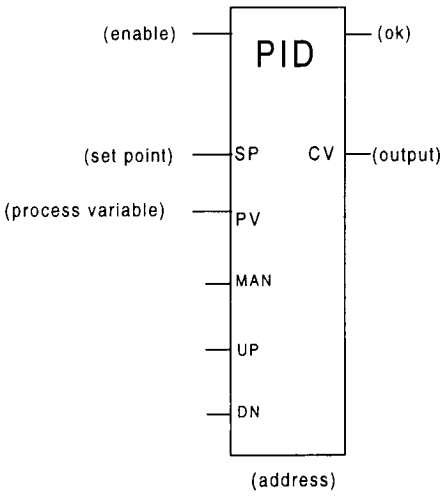
Blok PID



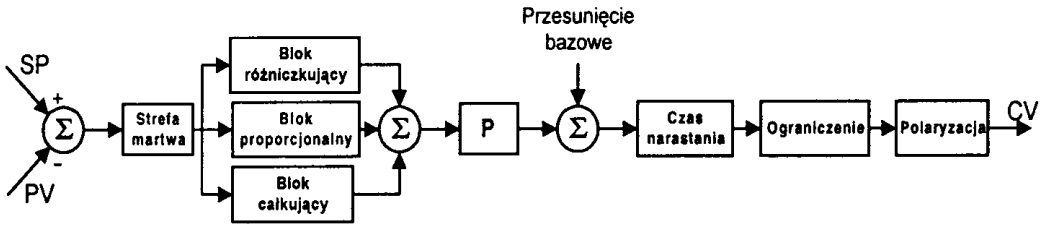
enable - EN - zezwolenie
ok - Potwierdzenie poprawności działania
SP - set point - wartość zadana
PV - process variable - zmienna procesowa
MAN - przełączanie trybu pracy automatyczna / ręczna
UP - inkrementacja wielkości wyjściowej w tr pracy MAN
DN - dekrementacja wielk. wyjściowej w tr. pracy MAN
CV - control variable - sygnał wyjściowy

•Omów okres próbkowania.
Odp. kolejne iteracje algorytmu PID są realizowane co podany okres z dokładnością do 100 mikrosekund. Jeżeli wartość ta jest równa 0, to kolejne iteracje są wykonywane w każdym cyklu, w którym uruchomiony jest blok PID. Zaleca się jednak ustawienie na 0.1s zamiast na 0 - algorytm PID będzie wykonywany w każdym cyklu obsługi ale nie mamy wtedy gwarancji że czas ten będzie stały.

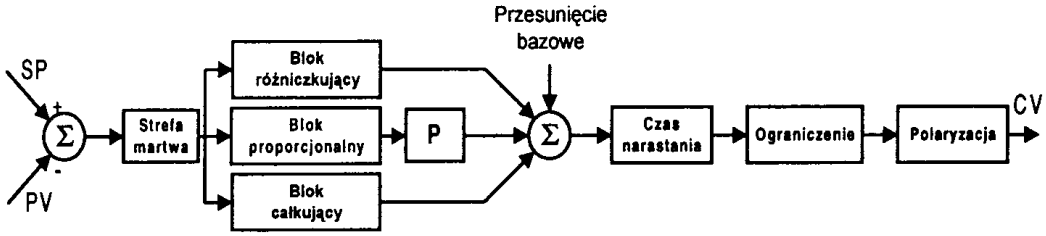
• PID-realizacja na Fanucu.
Odp. pytanie dodatkowe bylo czy adresy zajete przez PID'a sa jakos chronione, trzeba odpowiedziec ze nie i ze trzeba uwazac przy adresowaniu kolejnych zmiennych. Jest to regulator dyskretny, a nie ciągły. Wynika to z samej budowy sterownika, który jest urządzeniem cyfrowym. Blok ten został zaimplementowany w sposób który gwarantuje, że w ciągu jednego cyklu będzie wykonany tylko jedna iteracja algorytmu. Wykorzystuje on 40 kolejnych rejestrów w pamięci. Należy zwrócić uwagę podczas alokowania tych rejestrów, gdyż alokuje się je podając jedynie adres pierwszego rejestru, a dostęp do nich nie jest blokowany i łatwo przez nieuwagę je nadpisać, co spowoduje błędy. Rejestry te zawierają ustawienia bloku PID jak i wewnętrzne zmienne. Można je zmieniać w programie za pomocą funkcji przesuwania danych takich jak MOVE, lub BLKMOVE, oraz poprzez wpisywanie wartości w okienku „Tuning parameters" dostępnym w menu kontekstowym po kliknięciu prawym przyciskiem myszki na blok regulatora.

• Omów sposób adresowania bloku PID.
Odp. parametrem bloku funkcyjnego PID jest adres pierwszego z rejestrów roboczych, PID posiada 40 rejestrów konfiguracyjnych takich jak: stałą różniczkowania, okres próbkowania.

• Na transmitancji pokazać różnice między PID ISA i PID IND Dlaczego lepszy jest PID IND?
Odp. PID ISA: G(s) = k (1 + 1/T*s + T*s)



PID IND: G(s) = k + alfa/s + beta*s



IND dlatego, że przy zmianie k, nie zmieniają się pozostałe parametry, a w ISA suma sygnałów z wyjść wszystkich członów jest poddawana wzmocnieniu.

• Algorytm PID pozycyjny i przyrostowy

Pozycyjny: $$s_n = K_p\left[e_n + \frac{1}{T_i}\sum_{j=-\infty}^{n} e_j T_p + T_d\frac{e_n - e_{n-1}}{T_p}\right]$$

Kp - współczynnik wzmocnienia, Ti - czas zdwojenia, Td - czas wyprzedzenia, Tp - okres próbkowania, e(t) - uchyb regulacji, n - numer próbki. Suma w czlonie całkujacym musi być skończona bo inaczej nie byloby mozliwosci wyregulowania układu.

Przyrostowy: $$\Delta s_n = K_p\left(y_{n-1} - y_n\right) + K_i\left(u_n - y_n\right) + K_d\left(2y_{n-1} - y_{n-2} - y_n\right)$$

$K_i = \frac{T_p}{T_i}K_p$, $K_d = \frac{T_d}{T_p}K_p$. W przyrostowym czas całkowania nie może być 0 bo tylko człon całkujący zawiera wartość zadaną un. trzeba uwazac przy pozycyjnym: suma w czlonie calkujacym musi byc skonczona bo inaczej nie byloby mozliwosci wyregulowania ukladu

• Czy z PID'a przyrostowego mozna zrobic PD
Odp. Nie nie da sie, bo czlon calkujacy w tym algorytmie to [(Kc*Ts/Ti]* e(t) gdzie: e(t)=w (t)-y(t) (e-uchyb [error], w-wartosc zadana, y-wyjcie), i w nim jako w jedynym jest przechwowywana wartosc zadana !!! (czyli wystepuje w nim czyste e(t))

• Omów zasadę realizacji bloku PID w sterownikach Fanuc z uwzględnieniem czasu próbkowania.

Sterowniki PLC

• Podaj typy danych występujących w sterownikach PLC.
Odp. podkreślić te, które są w laboratoryjnym fanucu. Należy również podkreślić typ REAL, mimo iż w Legierskim go nie ma (w laboratorium wręcz przeciwnie).
Odp:
- BOOL – dane logiczne
- BYTE – bajt (8 bitów)
- SINT – liczba całkowita 8 bitowa
- INT – liczba całkowita 16 bitowa
- DINT – całkowita 32 bitowa
- LINT – całkowita 64 bitowa
- U.... gdzie ... to SINT, INT, DINT, LINT, liczby całkowite odpowiednio bitowe bez znaku
- REAL – liczba rzeczywista
- LREAL – liczba rzeczywista 64 bitowa
- TIME – czas trwania
- DATE – data

- TIME_OF_DAY – godzina dnia
- DATE_AND_TIME – data i czas
- STRING – ciąg znaków o zmiennej długości
- WORD – słowo (16 bitów)
- DWORD – ciąg 32 bitów (słowo podwójne)
- LWORD – ciąg 64 bitów (słowo długie)

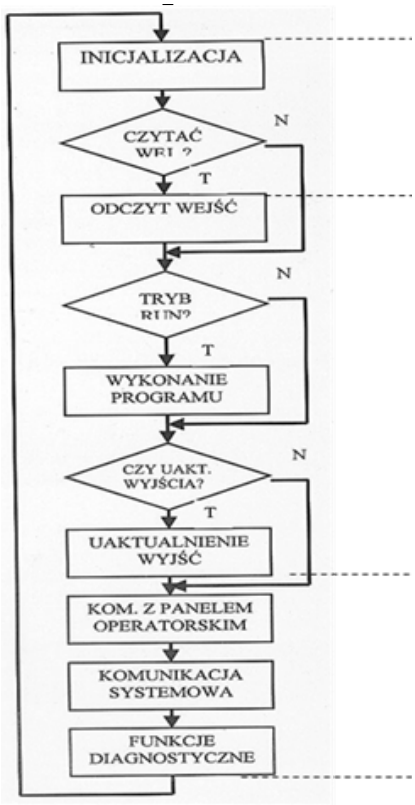
• Które z tych typow sa w GE Fanucu?
Odp. INT (16 bitowa), DINT (32 bitowa), BIT, BYTE, WORD, BCD-4 (czterocyfrowa liczba dziesiętna w formacie BCD - four digit binary coded decimal - liczba dziesiętna, której każda z 4 cyfr jest kodowana binarnie na 4 bitach - zakres 0-9999), REAL.

• Podaj typy danych przy budowie sterowników PLC
Odp: Intiger, char.

• Wymień cykl programowania sterowników
• Cykl (czyli te 7 elementow i kazal mi napisac ktore z tych elementow nie zawsze sie wkonuja.
• Pytanie przy sprawdzaniu czy czas cyklu jest staly?
Odp. Nie
Odp. Typowy cykl programowy sterownika składa się z następujących faz:

1. Inicjacja cyklu,
2. Czytanie sygnałów wejściowych, ------I
3. Wykonanie programu użytkownika, I To - T odpow.
4. Aktualizacja sygnałów wyjściowych, --I
5. Transmisja danych,
6. Komunikacja systemowa,
7. Wykonanie funkcji diagnostycznych.

1-7 - czas cyklu



• Cykl pracy sterownika PLC i zaznaczenie czasu odpowiedzi i czasu cyklu.

• Co do cyklu pytał jeszcze które rzeczy nie zawsze się wykonują?
Odp. Chodzi o bodajże 2, 3 i 4, czyli czytanie wejść, wykonanie programu i ustawienie wyjść. Powodem może być to że chcemy np zczytywać dane co sekundę a sterownik wiadomo szybciej działa.

• Schemat blokowy cyklu.
Odp. zrobiłem niepełny, więc pytanie dodatkowe: "Czy czas trwania cyklu jest stały?". Po chwili namysłu mówię, że jeśli się ustali czas próbkowania to jest stały - on na to, że nie do końca, ale że GE Fanuc to wyjątek i coś w tym stylu. Generalnie nie wiem jak jest w końcu, ale chyba cykl nie trwa zawsze tyle samo, poza "wyjątkami".

• Co to jest Min Slew Time i dlaczego sie go stosuje?
Odp. Chodzi o to, by w pewnych sytuacjach nie zniszczyć obiektu regulacji gwałtownym wzrostem sygnału sterującego. Przyda się również powiedzieć jak konfiguruje się ilość poszczególnych wejść i wyjść binarnych, bo standardowo nie jest ustawione tak jak być powinno (Rene raz nie sprawdzał tego, ale kazał skonfigurować).

Odp. Jest to minimalny czas zmiany sygnału sterującego z 0% na 100% Chodzi o to, by w pewnych sytuacjach nie zniszczyć obiektu regulacji gwałtownym wzrostem sygnału sterującego.

Język schematów drabinkowych LD

Język schematów drabinkowych LD należy do grupy języków graficznych i umożliwia realizację zadania sterowania za pomocą standaryzowanych symboli graficznych. Symbole się w obwodach w sposób podobny do szczebli (rungs). Obwód (network) jest definiowany jako zbiór wzajemnie połączonych elementów graficznych. Z obwodem może być skojarzona etykieta (label). Obwód LD ograniczony jest z lewej i prawej strony przez szyny prądowe (power rails). Szyny te nie są elementami obwodu. Prawa szyna może być rysowana w sposób jawny lub pozostawać w domyśle.
Języki graficzne są używane do przedstawienia przepływu odpowiedniej wielkości przez kolejne obwody reprezentujące pewną strategię sterowania. W przypadku języka LD wykonywanie programu polega na „przepływie prądu". Przepływ prądu następuje z lewej strony do prawej, przy czym obowiązują następujące zasady:

- wartość żadnego elementu obwodu nie powinna być wyznaczana dopóki nie wyznaczono wartości wszystkich jego wejść,
- wyznaczanie wartości elementu obwodu nie może być zakończone dopóki nie wyznaczono wartości dla wszystkich jego wyjść,
- wykonywanie programu dla całego obwodu nie jest zakończone dopóki nie wyznaczono wartości wyjść dla wszystkich elementów tego obwodu,
- w ramach elementu oprogramowania napisanego w języku LD kolejne obwody powinny być wyznaczane w kolejności z góry na dół, tak jak pojawiają się w schemacie drabinkowym, z wyjątkiem gdy kolejność ta ulega zmianie z powodu wprowadzenia elementów kontrolnych.

Elementy obwodu mogą być łączone poziomo lub pionowo. Stan elementéw łączących oznacza się jako ON lub OFF, odpowiednio do wartości logicznych 1 i 0. Stan lewej szyny powinien być uważany za ON, o ile nie występuje ona w nieaktywnym etapie grafu SFC. Nie definiuje się stanu dla prawej szyny. Połączenie poziome przekazuje stan elementu znajdującego się bezpośrednio po lewej do elementu po prawej. Stan połączenia pionowego odpowiada sumie logicznej OR dla stanów połączeń poziomych występujących po lewej stronie połączenia pionowego, tzn.:

- jeżeli wszystkie stany połączeń poziomych po lewej stronie połączenia pionowego są OFF to stan tego połączenia jest także OFF,
- jezeli przynajmniej jedno połączenie poziome po lewej stronie połączenia pionowego jest w stanie ON to połączenie pionowe jest także w stanie ON.

Stan połączenia pionowego powinien być przekazany wszystkim dołączonym po prawej stronie połączeniom poziomym, natomiast nie może być przekazywany do jakiegokolwiek elementu po lewej stronie połączenia pionowego.
Styk (contact) jest elementem przekazującym do połączenia poziomego po prawej stronie styku stan będący wynikiem mnożenia logicznego AND stanu linii łączącej po lewej stronie styku oraz wartości przypisanej mu logicznej zmiennej wejściowej, wyjściowej lub pamięciowej. **Styk nie modyfikuje wartości skojarzonej z nim zmiennej.**

***

---| |--- **Styk zwierny** (normalnie otwarty) - stan połączenia z lewej strony jest przenoszony na prawą, jeżeli skojarzona z nim zmienna logiczna (***) ma wartość 1. W przeciwnym razie prawe połączenie jest w stanie OFF.

***

---| / |--- **Styk rozwierny** (normalnie zamknięty) - stan połączenia z lewej strony jest przenoszony na prawą, jeżeli skojarzona z nim zmienna logiczna (***) ma wartość 0. W przeciwnym razie prawe połączenie jest w stanie OFF.

***

---| P |--- **Styk reagujący na zbocze narastające** - połączenie z prawej strony jest w stanie ON w czasie jednego wykonania, jeśli połączenie z lewej strony jest w stanie ON a skojarzona zmienna logiczna zmianiła wartość z 0 na 1. Poza tym stan połączenia z prawej strony jest OFF.

***

---| N |--- **Styk reagujący na zbocze opadające** - połączenie z prawej strony jest w stanie ON w czasie jednego wykonania, jeśli połączenie z lewej strony jest w stanie ON a skojarzona zmienna logiczna zmianiła wartość z 1 na 0. Poza tym stan połączenia z prawej strony jest OFF.

Cewka (coil) przekazuje stan połączeń z lewej strony na prawą bez zmian, powodując jednocześnie zapamiętanie stanu połączenia po swej lewej stronie przez przypisaną jej zmienną logiczną. Oprócz cewek standardowych, występują tu również cewki z zapamiętaniem stanu (retentive coils). Działanie tych cewek jest podobne jak ich standardowych odpowiedników, ale z taką cewką kojarzona jest zmienna zachowywana (retentive variable) bez konieczności użycia deklaracji VAR RETAIN. Oznacza to, że wartość zmiennej przypisanej do takiej cewki jest zapamiętywana po zatrzymaniu zasobu w którym jest zdefiniowana i odtwarzana po jego ponownym uruchomieniu.

***

---( )--- Cewka - stan połączenia z lewej strony cewki jest przenoszony na prawą stronę i zapamiętywany w skojarzonej zmiennej logicznej.

***

---(/)--- Cewka negująca - stan połączenia z lewej strony cewki jest przenoszony na prawą stronę a jego odwrotność jest zapamiętywana w skojarzonej zmiennej logicznej.

***

---(S)--- Cewka ustawiająca - skojarzona zmienna przyjmuje wartość 1 jeżeli połączenie z lewej strony jest w stanie ON i nie zmieni się aż do wyzerowania przez cewkę kasującą.

***

---(R)--- Cewka kasująca - skojarzona zmienna przyjmuje wartość 0 jeżeli połączenie z lewej strony jest w stanie ON i nie zmieni się aż do ustawienia przez cewkę ustawiającą.

***

---(P)--- Cewka reagująca na zbocze narastające - skojarzona zmienna przyjmuje wartość 1 na czas jednego wykonania jeśli połączenie z lewej strony zmieniło stan z OFF na ON. Stan połączenia jest zawsze przenoszony na prawą.

***

---(N)--- Cewka reagująca na zbocze opadające - skojarzona zmienna przyjmuje wartość 1 na czas jednego wykonania jeśli połączenie z lewej strony zmieniło stan z ON na OFF. Stan połączenia jest zawsze przenoszony na prawą.

***

---(M)--- Cewka z zapamiętaniem stanu (retentive coil).

***

---(SM)--- Cewka ustawiająca z zapamiętaniem stanu (set retentive coil).

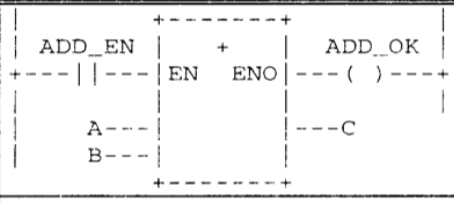
***

---(RM)--- Cewka kasująca z zapamiętaniem stanu (reset retentive coil).

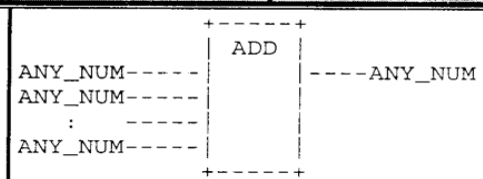
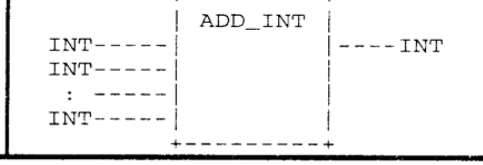
W języku LD mogą takze występować standardowe funkcje i bloki funkcyjne, z następującymi wyjątkami:

- parametry aktualne funkcji mogą być opcjonalnie pokazane przez wypisanie odpowiednich danych lub zmiennych na zewnątrz bloku, w sąsiedztwie nazw parametrów formalnych umieszczonych wewnątrz bloku,
- każdy blok powinien zawierać przynajmniej jedno wejście i wyjście logiczne by umożliwić przepływ prądu przez blok.

Dla celów programowania sterowników PLC **funkcję** (function) definiuje się jako element organizacyjny programu, który z chwilą wykonania dostarcza jeden element danych, chociaż może składać się on z wielu wartości, np. tablica. Funkcje nie zawierają wewnętrznej informacji o stanie (jest to więc „element statyczny"), tzn. wywołanie funkcji z tymi samymi argumentami (parametrami wejściowymi) zawsze daje tę samą wartość na wyjściu, w przeciwieństwie do bloku funkcyjnego.
W przypadku języka schematów drabinkowych LD wprowadza się do funkcji dodatkowe wejście logiczne EN (Enable - zezwolenie) oraz wyjście logiczne ENO (Enable Output). Użycie tych zmiennych podlega następującym regułom:
- Jeżeli w trakcie wywołania funkcji na wejściu EN występuje wartość FALSE (0) to operacje definiowane przez ciało funkcji nie będą wykonane a na wyjściu ENO także pojawia się wartość FALSE (0).
- Jeżeli na wejściu EN występuje wartość TRUE (1) to operacje zdefiniowane przez cialo funkcji są wykonywane a na wyjściu ENO powinna pojawić się wartość TRUE pod warunkiem, że operacje te zostały wykonane prawidłowo. Wystąpienie błędu wykonania powoduje pojawienie się na wyjciu ENO wartości FALSE (0).

Lp.	Opis	Przykład
1	Język *LD* – użycie *EN* i *ENO* jest obowiązkowe.	

Funkcje standardowe mogą być realizowane dla argumentéw wejściowych określonego typu (tzw. typed functions) albo dla argumemów pewnego rodzaju (tzw. overloaded function), np. funkcja dodawania ADD może być określona tylko dla argumentów całkowitych typu INT albo dla jakichkolwiek danych liczbowych, a więc typu LREAL, REAL, DINT, INT lub SINT. Funkcje definiowane przez użytkownika mogą być tylko dla argumentéw określonego typu.

Lp.	Opis	Przykład
1	Funkcja dla danych pewnego rodzaju (*overloaded function*), tu dla danych liczbowych rodzaju *ANY_NUM* (por. **2.2.2**)	
2	Funkcja dla danych określonego typu (*typed function*), tu dla danych typu *INT*.	

Tab. 2.18 Przykłady funkcji dla danych pewnego rodzaju i danych określonego typu.

Blok funkcyjny (function block), w przeciwieństwie do funkcji z chwilą wykonania może dostarczać na wyjściu jedną lub wiele wartości. Blok posiada wewnętrzne zmienne zawierające pewną informację o stanie bloku (jest więc „elementem dynamicznym").

Wszystkie wartości zmiennych wyjściowych oraz konieczne wartości zmiennych wewnętrznych są przechowywane pomiędzy kolejnymi chwilami wykonania bloku funkcyjnego, stąd wywołanie bloku z tymi samymi argumentami wejściowymi niekoniecznie musi prowadzić do tych samych wartości wyjściowych. W programie bloki funkcyjne mogą być wykorzystywane wielokrotnie (nazywane jest to ukonkretnieniem - ang. instance), a każda taka „kopia" bloku musi zawierać swoją nazwę oraz strukturę danych zawierających wyjście i wewnętrzne zmienne bloku.

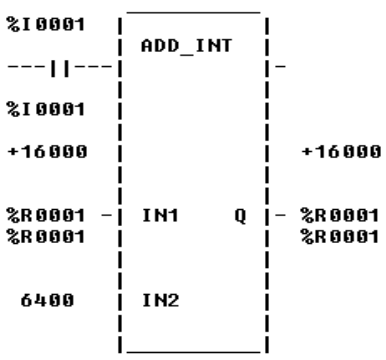
- Różnica między funkcją a blokiem funkcyjnym? Podaj przykłady w odniesieniu do GE FANUC.
- Jaka jest różnica między funkcją i blokiem funkcyjnym oraz przykłady jednych i drugich w Fanuc'u:

Odp. Funkcja - element STATYCZNY. Przykład funkcji - bloczek z funkcją matematyczną np. ADD, MUL. Wyjście z funkcji dla takiego samego wejścia jest TAKIE SAMO. Blok funkcyjny - element DYNAMICZNY. Wyjście z bloku ZALEŻY OD STANU WEWNĘTRZNEGO tego bloku (przeliczenie danych odbywa się w zadanych cyklach).

- Zasady uzycia blokow funkcyjnych i funkcji w jezyku drabinkowym.

- Jaka jest "podstawowa" różnica między stykiem, a cewką?

- Rozrysować w postaci struktury jezyka LD jak można zrealizować sumator. (warunkowy sumator to jak dasz bloczek Add_int , dajesz mu na wwejscia dwa inty, a na enable dajesz styk ktorym mozesz kierowac.) ---------->



- Cewka z zapamiętaniem zapamiętuje swój stan, nawet po wyłączeniu zasilania CPU i po właczeniu trzyma stan przez jeden cykl.

- Jaka jest realizacja bloków i funkcji w schemacie drabinkowym?

- Jak podać 50% sygnału i 20% skoku?

Odp: 50% - za pomocą bloku MOVE_INT, na wejście na stałe ustawić 16000 a na wyjściu jakiś rejestr, 20% - za pomocą bloku ADD_INT (czy coś takiego) na jednym wejściu rejestr z wyjścia 50% a na drugim wejściu 6400 (20% z 32000), wyjście do tego samego rejestru z, którego pobieramy dane. Wskazówka: blok MOVE_INT z rungu 1, blok ADD_INT w 2, PID w trzecim (na jego wejście PV rejestr).

- Jak zmienić wejście binarne za pomocą cewki?

Odp. niby się nie da, ale miszcz Rene mówi że się da cza tylko wiedzieć jak.

- Jakiej zmiennej wartość przypisuje cewka?

Odp. Zmienna binarna / logiczna - BOOL.

Konfiguracja sprzętu

- Co to jest konfiguracja sprzetu?

Odp. Jest to ustawianie pewnych parametrów ustawiających pracę modułów oraz adresów wykorzystywanych przez moduły systemu. Konfigurowanie ma na celu, między innymi, poinformowanie srerownika o modułach jakie zostały zainstalowane w kolejnych

gniazdach płyty łaczeniowej i jakie adresy logiczne zostały przypisane fizycznym wejściom i wyjściom.

• Omów programową konfigurację sterownika.
Odp. W większości sterowników wykorzystywana jest konfiguracja tworzona za pomocą oprogramowania do konfigurowania, przesyłana do jednostki centralnej z programatora. Moduł interfejsu komunikacyjnego pamięta konfigurację utworzoną za pomocą oprogramowania przy wyłączonym zasilaniu. Po zapamiętaniu konfiguracji w jednostce centralnej CPU będzie automatycznie konfigurowany po włączeniu zasilania.
Oprogramowanie do konfigurowania pozwala:

I. Utworzyć nowy projekt.
II. Zapisać konfigurację do modułu interfejsu komunikacyjnego sieci.
III. Wczytać istniejącą konfigurację z modułu interfejsu komunikacyjnego sieci.
IV. Porównać konfigurację zapisaną w module interfejsu komunikacyjnego z plikiem konfiguracyjnym zapisanym na komputerze.
V. Wykasować wygenerowaną automatycznie konfigurację, przesłaną poprzednio do modułu interfejsu komunikacyjnego.

Jednostka centranla zapamiętuje konfigurację programową w swojej nie ulotnej pamięci RAM. Zapisanie konfiguracji powoduje wyłączenie automatycznego konfigurowania, na skutek czego sterownik nie usunie wprowadzonej z programatora konfiguracji w czasie kolejnych uruchomień.
Wykasowanie konfiguracji z programatora powoduje jednak automatyczne wygenerowanie nowej konfiguracji. W takim przypadku, automatycznie wygenerowana konfiguracja obowiązuje do momentu ponownego przesłania konfiguracji z programatora.
Jednym z parametrów ustawianych z poziomu oprogramowania do konfigurowania jest parametr decydujący czy jednostka centralna po włączeniu zasilania odczytuje konfigurację z pamięci Flash, czy też odczytuje nową konfigurację i program z pamięci RAM. Jeżeli wybrana zostanie pamięć Flash, po włączeniu zasilania sterownika, jednostka centralna odczytuje konfigurację uprzednio zapisaną w pamięci Flash. Jeżeli wybrana zostanie pamięć RAM, po włączeniu zasilania sterownika, jednostka centralna odczytuje konfigurację i program sterujący z pamięci RAM.

• Jeśli chodzi o typ CPU oraz typy modułów współpracujących, to:
Odp. Jedyne konfiguracje sprzętowe, jakie trzeba wykonać, to ustawić liczbę wejść i wyjść dyskretnych (bo domyślnie jest źle ustawione). Rene nie sprawdzał tego.

• Omów konfigurację GE FANUC?

• Omów dane systemowe?

• Omów pliki statusowe w Gefanucu (always on)

• Jak ustawić liczbę wejść i wyjść dyskretnych?

Adresowanie w GE FANUC

• Adresowanie zmiennych w urzędzeniu o nazwie FANUC.
Odp. Wystarczy zapisać że piszemy procent potem typ np I (wejścia), Q (wyjścia), T (zm. Wielokrotnie wykorzystywane w programie), M (zm. Wewnętrzne), S (systemowe), G (globalne) potem adres.

• Pytanie dodatkowe: jak należy zaadresować DINT'a albo REAL'a?
Odp. podajemy adres rejestru, a fanuc zapisze wartość w dwóch kolejnych blokach 16-bitowych.

• Czy adresy zajete przez PID'a sa jakos chronione?
Odp. nie, trzeba uwazac przy adresowaniu kolejnych zmiennych.

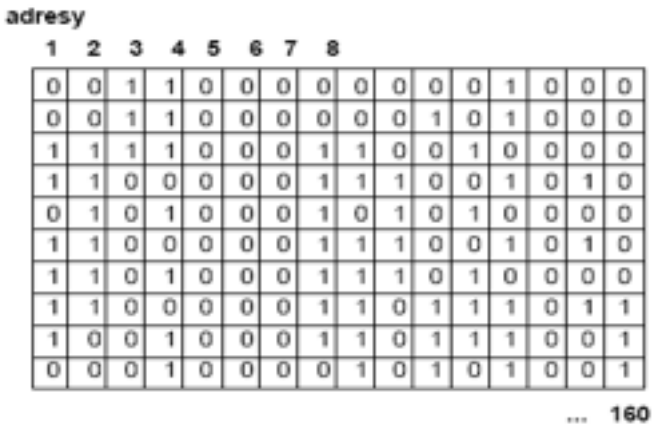
• Opisz bity systemowe.
Odp. Adresy typu %I, %Q, %M i %G posiadają powiązane z nimi bity zmiany i blokady wartości. Adresy typu %T, %S, %SA, %SB i %SC posiadają wyłącznie bity zmiany wartości. Bity zmiany wartości wykorzystywane są przez jednostkę centralną w przypadku przekaźników uaktywnianych zboczem sygnału. Po ustawieniu wartości bitu blokady na 1, można ją zmienić wyłącznie za pomocą programatora.

• Omów budowę pliku systemowego
Odp. Sprowadza się to do wymienienia typów zmiennych systemowych, %S, %SA, %SB, %SC , i mniej więcej tego co się w nich znajduje, np w %S: ALW_ON, podstawa czasu generatora fali prostokątnej. W %SA, %SB, %SC są głównie informacje o błędach. Poza tym ważne jest, że zmienne %S są tylko do odczytu. Dokładniej jest to opisane w manualu do Fanuca.

I. %S, %SA, %SB i %SC mogą być wykorzystywane bez względu na rodzaj styku.
II. %SA, %SB i %SC mogą być wykorzystywane z przekaźnikami z pamięcią styku.
III. %S mogą być wykorzystywane jako parametry wejściowe funkcji lub bloków funkcyjn.
IV. %SA, %SB i %SC mogą być wykorzystywane jako parametry wejściowe lub wyjściowe funkcji lub bloków funkcyjnych.

Odp. %S - tylko do odczytu, %SA, %SB, %SC - odczyt/zapis. Przechowują informacje na temat błędów, czasu itp. Wystarczyło podać nazwę i opis kilku wybranych. Ja podałem 4 i starczyło. Po co można edytować bity %SA, %SB i %SC -> Po wykryciu i usunięciu sygnalizowanego błędu zmienia się wartość odpowiedniego bitu z powrotem na 0.

• Omów zasadę przydziału pamięci przy adresowaniu układu GE FANUC.
Odp. Każdy adres pamięci bitów wskazuje na obszar zajmowany przez 1 bit. Format reprezentacji danych W pamięci bitów pokazano na rysunku poniżej. Przykład ten zawiera 160 indywidualnie adresowanych bitów, bit o adresie 1 znajduje się w lewym górnym rogu, a bit o adresie 160 W prawym dolnym.



Sterownik korzysta z sześciu typów adresów w pamięci bitów:
I.%I Standardowo ten obszar pamięci wykorzystywany jest przez wejścia dyskretne, a jego zawartość pokazuje tabela stanu wejść.
II.%Q Standardowo ten obszar pamięci wykorzystywany jest przez adresy wyjść, a jego zawartość pokazuje tabela stanu wyjść. Adresy typu %Q mogą posiadać pamięć stanu lub nie, w zależności od charakteru ich wykorzystywania w programie.
III.%M Standardowo ten obszar pamięci wykorzystywany jest przez zmienne wewnętrzne. Adresy typu %M mogą posiadać pamięć stanu lub nie, w zależności od charakteru ich wykorzystywania w programie.

IV. %T Standardowo ten obszar pamięci zajmowany jest przez zmienne, wielokrotnie wykorzystywane w programie. Stan zmiennych typu %T nie jest zapamiętywany po wyłączeniu zasilania lub po sekwencji trybów RUN-STOP-RUN. Zmienne typu %T nie mogą być wykorzystywane w przekaźnikach z pamięcią.
V. %G Obszar pamięci wykorzystywany przez dane globalne. Wartość danych przechowywanych w pamięci %G jest pamiętana po wyłączeniu zasilania. Adresy typu %G mogą być wykorzystywane z stykami i przekaźnikami z pamięcią, ale nie mogą być wykorzystywane w przypadku przekażników bez pamięci.
VI. %S Zmienne systemowe posiadające ściśle określone znaczenie.

• Adresowanie zmiennych w urzędzeniu o nazwie FANUC.
Odp. należało opisać zmienne rejestrowe:
%AI - wejście analogowe,
%AQ - wyjście analogowe,
%R - rejestr - komórka pamięci.
Binarne:
%I - wejście binarne - fizyczne wejście dwunastawowe, przechowywana w tablicy stanu wejść, uaktualnianej podczas fazy odczytu wejść w cyklu pracy sterownika. Może być przyporządkowana stykowi.
%Q - wyjście binarne - fizyczne wyjście dwunastawowe, przechowywana w tablicy stanu wyjść, uaktualnianej podczas fazy odczytu wyjść w cyklu pracy sterownika. Może być przyporządkowana cewce lub stykowi.
%M, %T - wewnętrzna zmienna binarna - reprezentuje wewnętrzną zmienną binarną programu sterującego. Może być przyporządkowana cewce lub stykowi.
%S - zmienne systemowe - reprezentują dane systemowe jak np. informacje o cyklu, błędach działania, aktualnym czasie itp. Mogą być przyporządkowane stykom, a niektóre z nich stykom lub cewkom.
%G - zmienne globalne - reprezentuja dane binarne wspólne dla kilku sterowników pracujących w sieci.

• Jak należy zaadresować DINT'a albo REAL'a?
Odp: podajemy adres rejestru, a fanuc zapisze wartość w dwóch kolejnych blokach 16-bit.

• Plik systemowy, co w nim się znajduje?
Odp. Chodziło o rzeczy ukryte pod adresami %S, np fala prostokątna 10ms, alw_on i co to jest %S można tylko odczytywać, czyli podstawiać pod styki, pytał które można edytować, jeśli się nie mylę chodziło o %SA, i chyba %SB.

• %R ile zajmuje tam jednostka pamięci?
• Jak zapisane są zmienne większe niż 1 bit?
• Ile standarowo jest pamięci przeznaczonej w rejestrze.
Odp. 1 słowo = 16 bitów.

• W versie pro otworzyć nowy folder - wpisać nazwę, nacisnąć przycisk dalej, przycisk zakończ pojawią się dwa okienka w okienku po prawej kliknąć kursorem dwa razy na nazwie Hardware Configuration pojawi sie nowe okno na którym znajduje się karta należy kliknąć dwa razy na nią pojawi się dalsze okno w którym znajdują się parametry tej że karty w katalogu Scan znajduje się parametr Sweep Timer - I teraz należy powiedzieć za co odpowiedzialny jest parametr Sweep Timer oraz co to jest Scan. Należy spodziewać się tego pytania jeżeli nie wie się jakie parametry są definiowan w czasie konfiguracji sterownika.