

## 6.2 Ćwiczenie 5b: Powiązanie markera i czujnika

- ❑ Taśmy uruchamia się do przodu po aktywacji **di\_Start** i gdy jest aktywny czujnik **di\_AtMiddle**. Zatrzymanie gdy jest aktywny **di\_Stop** lub jest aktywny **di\_AtRight**.

Uwaga! Ruch do przodu nie jest możliwy, jeżeli jest aktywny ruch taśmy do tyłu.

- ❑ Co się dzieje, gdy zatrzymamy transport (jak produkt jest pomiędzy czujnikami?)
- ❑ Wykorzystanie komórki pamięci, aby przechować informację o obecności produktu na linii. Możliwość wznowienia sekwencji
- ❑ Wykorzystanie komórki pamięci **m\_FactoryRunning** o adresie **M11.0** (typ BOOL), aby w szybki sposób zresetować wszystkie zmienne oraz wyjścia.

### 6.2.1 Rozwiązanie

```
IF "di_AtRight" OR (NOT "m_FactoryRunning") THEN
  "m_IsItemInLine" := FALSE;
ELSIF "dq_Forward" THEN
  "m_IsItemInLine" := TRUE;
END_IF;

IF (NOT "di_Stop") OR "di_AtRight" OR "dq_Backword" OR (NOT "m_FactoryRunning") THEN
  "dq_Forward" := FALSE;
ELSIF "di_Start" AND ((NOT "di_AtMiddle") OR "m_IsItemInLine") THEN
  "dq_Forward" := TRUE;
END_IF;
```

### 6.2.2 Omówienie kodu programu

Komórka pamięci fabryka uruchomiona została przygotowana w celu ręcznego resetu zmiennych wchodzących w skład kodu programu. W środowisku programistycznym jest możliwość ręcznego zapisania wartości do pamięci. Wykorzystując tę funkcjonalność oraz używając zmiennej czy fabryka jest uruchomiona w zapisach związanych z warunkami na wyłączenie, możemy wszystkie zmienne wykorzystywane w całym kodzie programu za pomocą kilku kliknięć myszką ustawić w stan 0, czyli zresetować.

Trzeba pamiętać o tym, aby tą komórkę pamięci dodawać wszędzie, gdzie piszemy warunki na reset zmiennych lub wyjść. Jest to potrzebne, aby testy danej funkcjonalności rozpoczynać zawsze od warunków początkowych, czyli wszystkie zmienne i wyjścia powinny mieć wartość 0. Dopiero w takiej sytuacji możemy sprawdzić, czy kod programu, który napisaliśmy, działa poprawnie.

Ręczna modyfikacja polega na tym, że włączymy monitoring kodu programu, czyli tak zwane "okularki". Wówczas możemy kliknąć prawym przyciskiem myszy na nazwę zmiennej w kodzie programu, czyli w tym wypadku na zmienną "fabryka uruchomiona", i z menu kontekstowego wybrać opcję "modyfikuj". Następnie wybieramy stan, który chcemy przesłać do pamięci, czyli w przypadku tej zmiennej będzie to albo 0, albo 1.

Drugi sposób, gdzie można zapisać wartość do pamięci, to oczywiście wykorzystanie tablic podglądu. Przed rozpoczęciem testów kodu programu powinniśmy tą zmienną ustawić w stan wysoki.

Pierwsza instrukcja warunkowa IF jest powiązana z obsługą komórki pamięci, czyli stanowi warunki, kiedy ta komórka pamięci przyjmuje wartość 1, a w jakiej sytuacji przyjmuje wartość 0. Ta pamięć została przygotowana w celu posiadania informacji, czy produkt jest na linii pomiędzy czujnikami w środku a prawym.

Jest ona potrzebna do wznowienia ruchu, gdy zostanie zatrzymana taśma w trakcie transportu, czyli gdy produkt będzie znajdował się pomiędzy czujnikami i zostanie wciśnięty przycisk Stop.

Warunek, jaki musi być spełniony, aby zapisać wartość 0 do komórki pamięci "Czy przedmiot jest na linii", to stan wysoki z czujnika po prawej stronie, czyli czy czujnik wykrywa produkt. Lub drugi składnik warunku, czyli sprawdzenie, czy komórka pamięci "fabryka uruchomiona" jest w stanie niskim. Więc jeżeli ten warunek (który jest zbudowany z użyciem bramki OR) jest spełniony, to wówczas mamy reset wartości w komórce pamięci. Wartość 1 zostanie zapisana do komórki pamięci w momencie, gdy nastąpi ruch taśmy do przodu.

Więc jeżeli żaden z warunków tej instrukcji warunkowej IF nie jest spełniony, związany z obsługą komórki pamięci, to sterownik realizuje pozostały kod programu.

Druga instrukcja warunkowa IF służy do sprawdzania warunków związanych z włączeniem i wyłączeniem ruchu taśmy do przodu.

Pierwszy warunek instrukcji warunkowej IF... ELSIF do sprawdzenia warunków na zatrzymanie ruchu taśmy do przodu. Więc ten warunek składa się z czterech składników, a przynajmniej jeden z nich musi być prawdą, aby nastąpiło zatrzymanie taśmy. Pierwszy z nich to aktywacja przycisku Stop, czyli jeżeli przycisk Stop zostanie wciśnięty, taśma się zatrzyma. Drugi składnik to pojawienie się produktu naprzeciw czujnika po prawej stronie, czyli jeżeli ten czujnik wykryje produkt, gdy taśma będzie jechała do przodu, wówczas taśma się zatrzyma. Trzeci składnik to sprawdzenie, czy jest wystawiona jazda do tyłu, więc to jest składnik związany z zabezpieczeniem, aby nie uszkodzić silnika. Czwarty składnik to sprawdzenie, czy komórka pamięci "fabryka uruchomiona" jest w stanie niskim.

Więc jeżeli jeden z tych czterech składników jest prawdą, następuje zapisanie wartości 0 do pamięci związanej z jazdą do przodu, i to powoduje zatrzymanie taśmy. Więc jeżeli warunek jest spełniony, następuje zatrzymanie taśmy, i sterownik przechodzi do realizacji kolejnej części kodu programu już poza instrukcją warunkową IF... ELSIF.

Jeżeli żaden ze składników na zatrzymanie nie jest prawdą, wówczas jest sprawdzany kolejny warunek, który stanowi warunek na włączenie jazdy taśmy do przodu.

Więc tutaj zmienił się warunek na włączenie taśmy do przodu, bo jednym z warunków jest jeszcze komórka pamięci mówiąca o tym, czy przedmiot znajduje się na linii, i ta komórka pamięci związana z obecnością przedmiotu na linii jest powiązana z czujnikiem na środku za pomocą bramki OR. Czyli warunki na uruchomienie taśmy to aktywacja przycisku START, czyli musi być wciśnięty przycisk Start, i musi znajdować się produkt naprzeciw czujnika środkowego. Więc to jest pierwszy scenariusz. Drugi scenariusz, gdzie może być uruchomiona jazda do przodu, to aktywacja przycisku START i produkt znajduje się pomiędzy czujnikami (czyli komórka pamięci związana z obecnością produktu pomiędzy czujnikami ma wartość 1). Czyli jeżeli jeden z tych scenariuszy zostanie spełniony, wówczas następuje uruchomienie jazdy do przodu.

## 7 Pamięć systemowa

W sterownikach PLC SIMATIC S7-1200 od firmy Siemens, pamięć systemowa odgrywa istotną rolę, umożliwiając odczyt informacji dotyczących przebiegów zegarowego bezpośrednio w kodzie programu. Dzięki niej jest możliwe sygnalizowanie awarii poprzez np. migotanie diody, na przykład diody niebieskiej. W niniejszym rozdziale dokładnie przeanalizujemy zasady funkcjonowania pamięci systemowej oraz przedstawimy praktyczne metody wykorzystania funkcji przebiegów zegarowych.

### 7.1 Ćwiczenie 6a: Pulsowanie sygnału wyjściowego

- Ruch taśmy do przodu gdy aktywny **di\_Start**
- Lampka **dq\_LedGreen** pulsuje z częstotliwością 1Hz, gdy jest ruch taśmy

#### 7.1.1 Rozwiązanie

```
"dq_Forward" := "di_Start";  
"dq_LedGreen" := "dq_Forward" AND "Clock_1Hz";
```

#### 7.1.2 Omówienie kodu programu

Uruchomienie jazdy do przodu zależy od wartości, która jest przechowywana w pamięci związanej z przyciskiem Start. Jeżeli w pamięci znajduje się wartość 1, to taśma będzie jechała do przodu, czyli wtedy, gdy przycisk Start jest wciśnięty. Jeżeli przycisk Start nie jest wciśnięty, wówczas w pamięci sterownika w obszarze wejść znajduje się wartość 0, która zostaje przypisana do wyjścia Jazda do przodu. W tej sytuacji taśma nie będzie się poruszać do przodu.

Oczywiście, tę linijkę kodu można byłoby zapisać za pomocą instrukcji warunkowej If, ale optymalniejszym rozwiązaniem jest wykorzystanie klasycznego przypisania.

Światło zielone sygnalizuje nam, czy jest aktualnie uruchomiona jazda przonośnika do przodu. Ta sygnalizacja polega na cyklicznym pulsowaniu zielonej lampy, które zostało zrealizowane na systemowym bicie, które zmienia się cyklicznie z częstotliwością jednego Hz.

Warunek, jaki musi być spełniony, żeby lampka pulsowała, to aktualnie musi być jazda przonośnika do przodu oraz stan bitu pamięci zegara 1 Hz powinien mieć wartość 1. Zgodnie z bramką AND, oba składniki muszą przyjąć wartość 1, tylko wtedy światło zostanie włączone.

Więc jeżeli będzie jazda do przodu, aktualnie trwała, a bit zegarowy 1 Hz będzie w stanie niskim, to światło będzie wyłączone. Innymi słowy, włączenie i wyłączenie światła będzie zależało od przebiegu zegarowego, gdy trwa jazda do przodu.

Tę linijkę kodu związaną ze światłem zielonym też można zapisać za pomocą instrukcji warunkowej If, ale lepszym rozwiązaniem jest klasyczne przypisanie.

## 7.2 Ćwiczenie 7a: Podział aplikacji na moduły

### Moduł 1

- Utworzenie bloku FC o nazwie **FC\_Conveyor**
- Ruch taśmy do przodu gdy aktywny **di\_Start** i aktywny **di\_Selector**

### Moduł 2

- Utworzenie bloku FC o nazwie **FC\_LightSygnalization**
- Lampka **dq\_LedGreen** pulsuje z częstotliwością 1Hz, gdy jest ruch taśmy
- Bezwarunkowe wywołanie bloków FC w bloku OB1(Main)
- Testy funkcjonalności (podgląd kodu programu w każdej funkcji FC)

#### 7.2.1 Rozwiązanie

FC_Conveyor
"dq_Forward" := "di_Start";

FC_LightSygnalization
"dq_LedGreen" := "dq_Forward" AND "Clock_1Hz";

Main
"FC_Conveyor"(); "FC_LightSygnalization"();

#### 7.2.2 Omówienie kodu programu

Kod programu można napisać na dwa sposoby. Pierwszy z nich to tzw. program liniowy, gdzie cały kod programu znajduje się w jednym bloku i zazwyczaj obejmuje dużą liczbę linii kodu. To utrudnia analizę i zmniejsza czytelność takiego kodu.

Lepszym rozwiązaniem jest drugi sposób, czyli podział programu na mniejsze fragmenty, a te fragmenty umieszczamy w funkcjach. W przypadku sterowników PLC, funkcją jest FC.

Liczba funkcji, jaką dodamy, zależy od wykonywanego zadania. Nawet w przypadku małych zadań należy podzielić kod programu. Sposób podziału zależy już od programisty.

Podział ten jest stosowany w celu zwiększenia czytelności projektu i ułatwienia pracy nad jego rozbudową w przyszłości.

Funkcja związana z sygnalizacją świetlną powinna zawierać kod programu, który bezpośrednio steruje światłkami lub diodami LED.

W funkcji FC\_LightSygnalization mamy następujące warunki: światło zielone sygnalizuje nam, czy aktualnie jest uruchomiona jazda przonośnika do przodu. Sygnalizacja ta polega na cyklicznym pulsowaniu zielonej lampy, zrealizowanym na systemowym bicie, które zmienia się cyklicznie z częstotliwością jednego Hz.