ST6-Realizer

część 2

Narysuj swój program!

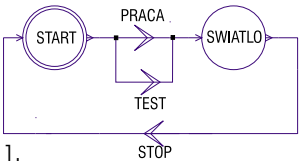
W drugiej części kursu przedstawiamy kompletny cykl projektowania, na przykładzie mikroprocesorowego regulatora temperatury, za pomocą Realizera. W programie sterującym pracą mikrokontrolera rezygnowaliśmy z wprowadzenia programowanej histerezy zapobiegającej naprzemiennemu włączaniu i wyłączaniu regulatora przy zadanej wartości stabilizowanej temperatury.

Aby przygotować projekt za pomocą Realizera należy:
- narysować schemat logiczny struktury projektowanego układu,
- zdefiniować algorytm działania programu za pomocą grafu.

Z doświadczenia wiem, że wielu początkujących elektroników stykając się po raz pierwszy z programem Realizera podczas tworzenia oprogramowania jakiegoś sterownika robi podstawowy błąd nie rysując grafu będącego odpowiednikiem algorytmu.

Co to jest ten algorytm?

Jest to po prostu opis działania programu, podzielony na stany pracy, do których przechodzi mikrokontroler pod wpływem zdarzeń (warunków), które mogą występować na zewnątrz lub wewnątrz mikrokontrolera. W programie Realizer zapisuje się algorytm graficznie, w postaci grafu przejść. Graf jest budowany z predefiniowanych elementów znajdujących się w bibliotekach Realizera, takich jak STATE (stan), CONDITION (warunek), INITIALSTATE (stan początkowy). Schemat logiczny programu jest ściśle powiązany z grafem, a do określania powiązań służą elementy: STATE INPUT (sygnały wejściowe dla stanu), STATE OUTPUT (sygnały wyjściowe w określonym stanie). Przykład najprostszego algorytmu zapisanego zgodnie z regułami Realizera przedstawiono na **rys. 1**.



Rys. 1.

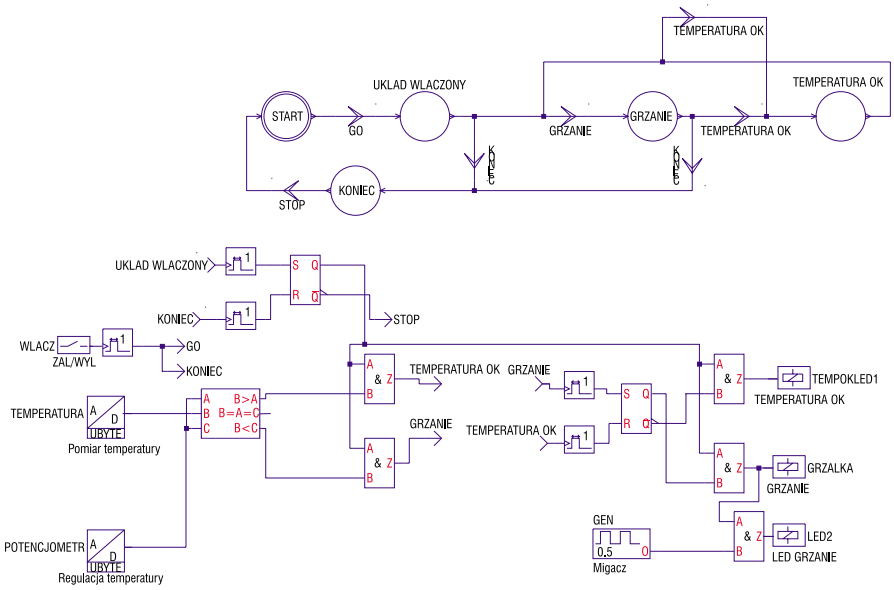
Aby ułatwić Czytelnikom poznawanie Realizera, oprzemy się na gotowym projekcie, prostego regulatora temperatury, przygotowanym przez autora.

Projekt przykładowy

Na początku opracowywania projektu musimy zadać sobie pytanie: co nasz układ powinien robić? Na pewno powinien mierzyć temperaturę, a wynik pomiaru powinien zostać porównany z wartością zadaną. W zależności od wyniku porównania układ powinien „zadecydować" czy włączyć grzałkę czy nie. Na **rys. 2** przedstawiono schemat logiczny układu wraz z grafem przejść dla Realizera.

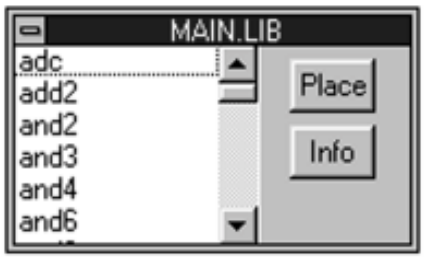
Schemat logiczny

Wszystkie elementy biblioteczne użyte w schemacie wchodzą w skład standardowej biblioteki MAIN LIB Realizera (**rys. 3**), do której dostęp jest możliwy po wciśnięciu przycisku pokazanego na **rys. 4**.



Rys. 2.

Tak więc, aby narysować prezentowany schemat wystarczy pobrać



Rys. 3.

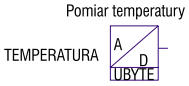
z biblioteki odpowiednie elementy, ułożyć je na planszy (arkuszu roboczym) i połączyć ze sobą. Możemy to zrobić następująco: podświetlamy myszką element o nazwie *Stateinit* i naciskając przycisk *Place* przemieszczamy element w wybranym miejscu strony. Następnie naciskamy prawy przycisk myszy, co powoduje otwarcie się okna *Edit the value*, w którym wpisujemy nazwę elementu np. START. Wybierając nazwę tego elementów nie należy używać polskich znaków, ponieważ nie będą one poprawnie wyświetlane przez program.

W ten sposób mamy umieszczony na planszy pierwszy element. Z biblioteki wybieramy kolejny element

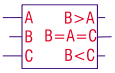

Rys. 4.

Rys. 5.

Rys. 6.



Rys. 7.



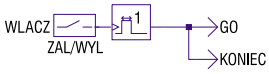
Rys. 8.

Rys. 9.

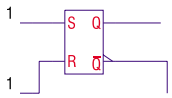
Rys. 10.

Rys. 11.

Rys. 12.



Rys. 13.



Rys. 14.

o nazwie *Condition,* który umieszczamy obok poprzedniego i również nadajemy mu nazwę, powiedzmy *PRACA*. Po naciśnięciu przycisku możemy przystąpić do połączenia tych elementów. Po najechaniu kursorem na końcówkę elementu pojawi się mały krzyżyk, który wskazuje końcówkę możliwą do dołączenia. W tym momencie naciskamy lewy przycisk myszy, co spowoduje, że rysowana linia zostaje „zaczepiona" na końcówce. Następnie prowadzimy linię do kolejnego elementu, aż ukaże się krzyżyk na jego końcówce. W tym momencie naciskamy ponownie lewy przycisk myszy co spowoduje, że pomiędzy dwoma elementami będzie zestawione połączenie. Podczas rysowania schematu zauważycie, że program automatycznie wybiera trasę przebiegu linii połączeniowej.

Następnie wybieramy z biblioteki element o nazwie *State* i rysujemy schemat grafu korzystając z poznanych elementów *State, Condition.* Jak widać na schemacie z rys. 2, zastosowaliśmy dwa wejścia analogowe z przetwornikiem A/D, jedno wejście cyfrowe DIGIN oraz trzy wyjścia DIGOUT.

Wejście analogowe *TEMPERATURA* jest wykorzystywane do pomiaru zmian napięcia na termistorze. Natomiast wejście analogowe *POTENCJOMETR* jest wykorzystywane przy pomiarze napięcia na potencjometrze zadającym próg załączenia grzałki.

Wejście cyfrowe DIGIN *ZAŁ/WYŁ* (**rys. 5**) wykorzystano jako włącznik inicjujący działanie programu. Wyjścia cyfrowe DIGOUT (**rys. 6**) sterują zewnętrznymi elementami takimi jak diody LED oraz przekaźnik załączający obwód grzałki.

Sygnały z przetworników analogowo-cyfrowych A/D (**rys. 7**), w postaci binarnych słów ośmiobitowych, są podawane szeregowo na wejście A, B, C komparatora (**rys. 8**). Następuje w nim porównanie wartości sygnału *TEMPERATURA*, przetworzonego przez przetwornik A/D do postaci cyfrowej, z wartością cyfrową na wyjściu przetwornika A/D *POTENCJOMETR*. Stan na wejściu A określa górny próg, a na wejściu C dolny próg zadziałania. W naszym przypadku wartość cyfrowa na obydwu wejściach jest jednakowa. Komparator posiada trzy wyjścia. W układzie wykorzystaliśmy dwa: wartość B jest mniejsza od C (B<C) i wartość B jest większa od A (B>A).

Graf przejść

Stworzenie kompletnego programu za pomocą REALIZERA wymaga jeszcze opisanie sposobu działania mikrokontrolera. Służy do tego graficznie zdefiniowany algorytm działania procesora, tzw. grafu, określający zależności logicznych pomiędzy zdarzeniami. Do tworzenia algorytmu w projekcie *REGULATOR TEMPERATURY* użyto następujących funkcji:

- INITIAL STATE - **rys. 9** (stan początkowy, czyli początek działania programu procesora),
- CONDITION - **rys. 10** (warunek),
- STATE - **rys. 11** (stan).

Z elementem CONDITION powiązany jest ściśle element STATE INPUT, którego symbol graficzny pokazano na **rys. 12**. Podczas rysowania programu każdy element CONDITION powinien mieć nazwę np. „GRZANIE" i taką samą nazwę powinien mieć element STATE INPUT. Tak samo jest z funkcją STATE. Ta funkcja jest związana z elementem STATE OUTPUT i obie powinny mieć takie same nazwy. Jak widać na schemacie (rys. 2) programu zasada ta została zachowana.

Działanie programu przebiega następująco: pojawienie się na wejściu *ZAŁ/WYŁ* stanu wysokiego powoduje wyzwolenie krótkiego impulsu na wyjściu modułu EDGE (**rys. 13**), a w konsekwencji na wejściu STATEIN *GO*. Powoduje to pojawienie się stanu wysokiego na STATEOUT *(UKLADWLACZONY)*, który podany na wejście S przerzutnika RS powoduje podanie logicznej jedynki na połączone wejścia A bramek AND.

Program oczekuje na następne zdarzenie. Spójrzmy na nasz algorytm: może to być zdarzenie *KONIEC*, *GRZANIE* lub *TEMPOK*. Zaistnienie zdarzenia *KONIEC* nastąpi

Rys. 15.

w wyniku ponownego podania na wejście ZAŁ/WYŁ stanu wysokiego. Na STATEIN *KONIEC* pojawi się krótki impuls. Obydwa sygnały STATEIN *KONIEC* i STATEIN *GO* są generowane na jednym wyjściu, jak to pokazano na rys. 13.

Mogłoby się wydawać, że układ po ponownym podaniu stanu wysokiego na wejście ZAŁ/WYŁ przejdzie w stan *GO*. Jednak to nie nastąpi, ponieważ program reaguje tylko na takie zdarzenia, jakie występują po stanie w jakim się aktualnie znajduje. Daje to nam możliwość generowania z jednego wejścia różnych sygnałów przez przechodzenie z jednego stanu do następnego.

Kolejnym interesującym nas stanem jest *GRZANIE*, którego wykonanie powoduje przejście programu w stan *GRZANIE*. Warunek *GRZANIE* zostanie spełniony w przypadku, gdy w wyniku porównania przez komparator wartości zadanej potencjometrem z wartością zmierzoną na termistorze, na wyjściu komparatora B<C pojawi się poziom wysoki. Przejście w stan *GRZANIE* powoduje, że na STATEOUT *GRZANIE* pojawi się poziom wysoki, który jednocześnie zostanie podany na wejście S przerzutnika RS (**rys. 14**). Na jego wyjściu Q pojawi się stan wysoki, który poprzez bramkę AND podany jest wyjście cyfrowe DIGOUT sterujące pracą grzałki. Równocześnie z wyjścia Z tej bramki AND sygnał zostaje podany na wejście A bramki, której wyjście Z steruje wyjściem cyfrowym zasilającym diodę LED. Drugie wejście B bramki AND sterowane jest z generatora, który generuje impulsy o czasie trwania 0,5s. Powoduje to, że kiedy program jest w stanie *GRZANIE*, dioda LED migocze, a w innych stanach dioda nie świeci. Gdy regulowana temperatura (podczas nagrzewania) wzrośnie do wartości wcześniej zadanej potencjometrem, zostaje spełniony warunek *TEMPOK* i program mikrokontrolera przechodzi w stan *TEMPOK*. Aby warunek ten został spełniony, wyniki pomiarów z przetworników A/D porównane przez komparator muszą dać wynik B>A, czyli zmierzona temperatura będzie wyższa od ustawionej.

Gdy na wyjściu B>A komparatora wystąpi poziom wysoki, to wystąpi on jednocześnie (poprzez STATEIN *TEMPOK* oraz STATEOUT *TEMPOK*) na wejściu R (zerującym) przerzutnika. Powoduje to pojawienie się poziomu niskiego na wyjściu mikrokontrolera DIGOUT *GRZANIE* i LED *GRZANIE*. Na wyjściu DIGOUT *TEMPOK LED* pojawi się poziom wysoki, w wyniku którego zapali się dioda *TEMPOK*. Następnie program mikrokontrolera oczekuje na ponowne spełnienie któregoś z warunków *GRZANIE*, *KONIEC* i tak w „kółko".

Ciąg dalszy

Teraz, gdy wiemy jak działa program, możemy przystąpić do dalszych czynności przy jego tworzeniu. Gdy mamy już wszystko narysowane, należy wszystkie wejścia analogowe oraz wejścia i wyjścia cyfrowe przypisać do fizycznych wyprowadzeń mikrokontrolera.

Kursorem najeżdżamy na wybrane wejście i dwa razy klikamy, w wyniku czego otwiera się okno *Hardware connections* (jak na **rys. 15**). Po jego lewej stronie znajduje się spis wolnych wyprowadzeń mikrokontrolera, a po prawej spis wykorzystanych.

Po zaznaczeniu w lewym oknie interesującego nas wejścia klikamy na przycisk *Connect* i wybrane wejście jest przenoszone do okna prawego. Po przeniesieniu należy koniecznie nacisnąć *Close*. W tym momencie zostanie przypisany sygnał ze schematu do fizycznego wyprowadzenia mikrokontrolera. Tak postępujemy kolejno ze wszystkimi wejściami i wyjściami. Może się zdarzyć, że chcemy już przypisane wyprowadzenie skonfigurować inaczej lub usunąć. W tym celu zaznaczamy wybrane wejście po prawej stronie okna i klikamy *Discon*.

W mikrokontrolerze ST62T01 jako wejścia cyfrowe mogą być wykorzystane wszystkie porty mikrokontrolera PA1..PB7. Jako wejście analogowe z przetwornikiem analogowo cyfrowym mogą być skonfigurowane tylko cztery wyprowadzenia: PB3, PB5, PB6, PB7. Po przypisaniu sygnałów

wejściowych i wyjściowych wyprowadzeniom, następnym krokiem jest przeprowadzenie analizy projektu. W tym celu należy na pasku menu wybrać *Analyse* i polecenie *Go* lub nacisnąć odpowiednią ikonę. Otwiera się okno, w którym obserwujemy cały proces analizy (**rys. 16**).

Rys. 16.

Podczas analizy tworzone są miedzy innymi wynikowy plik HEX do programowania mikrokontrolera oraz plik raportu, w którym opisano skonfigurowane wyprowadzenia oraz ilość wykorzystanej pamięci mikrokontrolera.

Kolejnym krokiem po skończonej kompilacji jest sprawdzenie programu na symulatorze, którym zajmiemy się za miesiąc.

Krzysztof Górski, AVT
krzysztof.gorski@ep.com.pl

Na płycie CD-EP2/2001B opublikowaliśmy ST6-Realizera w pełnej wersji funkcjonalnej. Jest on także dostępny (wraz z katalogiem procesorów ST62) na płycie CD-EP2.