Systemy wbudowane

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**

- Systemy wbudowane z reguły są systemami czasu rzeczywistego
- W programowaniu systemów czasu rzeczywistego korzysta się albo z
  - systemów operacyjnych czasu rzeczywistego

  albo ze
  - specjalnych algorytmów działania systemu
    - np. algorytmy synchroniczne (w prostszych systemach; zaleta – mniejsze wymagania dotyczące zasobów pamięci)

Sterowanie dwustanowe najczęściej sprowadza się do wykonania pewnej sekwencji operacji, przy czym przejście od jednej operacji do drugiej następuje po spełnieniu określonych warunków, np. osiągnięcie określonego celu lub upływu zadanego czasu. W takim przypadku mamy do czynienia ze sterowaniem sekwencyjnym. W wielu pojawiają się procesy współbieżne. We wprowadzeniu pokazano sposób tworzenia algorytmów synchronicznych dla wybranych przykładów sterowania dwustanowego.

Systemy wbudowane

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**

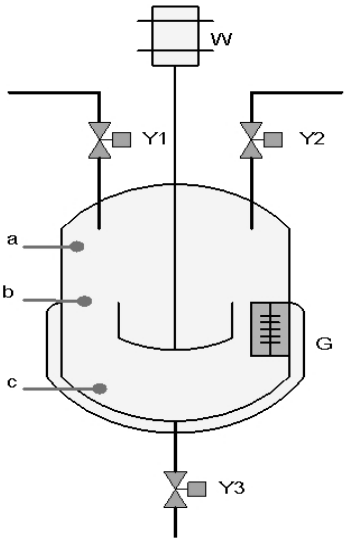
Sterowanie mieszalnika

a, b, c - czujniki poziomu
(1 – czujnik zanurzony,
0 - czujnik wynurzony)

Y1, Y2, Y3 - zawory sterowane

W - sterowanie mieszadła

G - sterowanie grzejnika



Zygmunt Kubiak

Programowanie systemów wbudowanych (2)

Sterowanie pracą mieszalnika

Prostym przykładem jest sterowanie pracą mieszalnika. Algorytm jest następujący:

•zakładamy, że zbiornik jest pusty;

•proces uruchamiamy naciśnięciem przycisku **S** (Start), co powoduje otwarcie zaworu **Y1** i napełnianie medium **1**;

•osiągnięcie poziomu **b**, powoduje zamknięcie zaworu **Y1**, otwarcie zaworu **Y2**, a więc dopływ medium 2 oraz uruchomienie mieszadła **W**;

•osiągnięcie poziomu **a**, powoduje zamknięcie zaworu **Y2** i włączenie na zadany czas grzejnika **G***;*

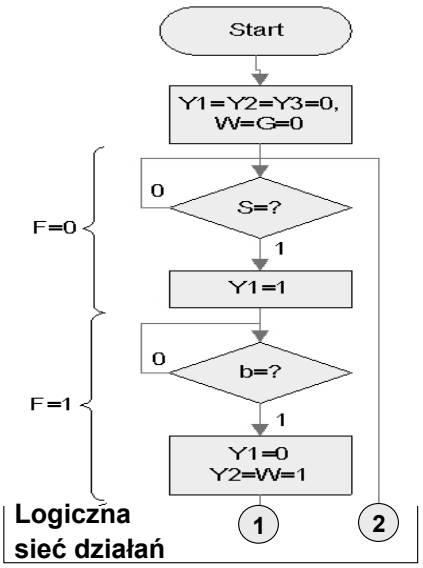
•po upływie czasu **t** następuje wyłączenie mieszadła **W** i grzejnika **G** oraz otwarcie zaworu **Y3**, opróżniającego zbiornik;

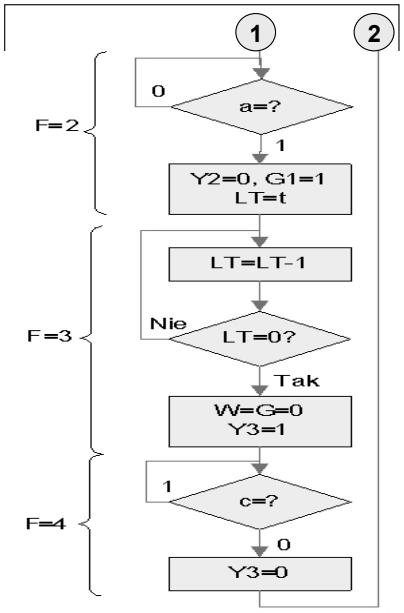
•gdy poziom cieczy spadnie poniżej czujnika **c**, następuje zamknięcie **Y3**;

•jeżeli przycisk **S** jest nadal wciśnięty, to cykl się powtarza.

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**

Sterowanie mieszalnika



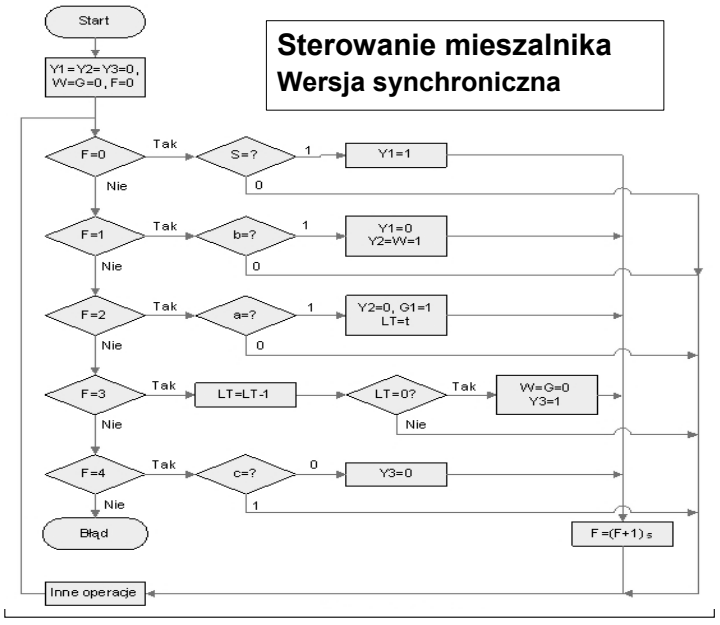


Sterowanie pracą mieszalnika

Sieć działań przedstawia logiczny przebieg procesu, który został podzielony na pięć kroków. W takiej postaci algorytm ten nie powinien być przeniesiony do postaci programu sterowania obiektu. W takim przypadku mikrokontroler, realizujący program zdecydowaną większość czasu zużywałby w pętli oczekiwania na zajście określonego zdarzenia, tzn. zmiana stanu czujnika lub upływ czasu. Jego moc obliczeniowa byłaby wykorzystana w nieznacznym stopniu. Stan nadzorowanego obiektu zmienia się bardzo wolno w stosunku do możliwości mikrokontrolera. Ponadto w takim przypadku trudno byłoby kontrolować poprawna pracę mikrokontrolera. Trudniej również taki program rozszerzać.

Systemy wbudowane

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**



Opracowanie Z.Kubiak na podstawie: Musierowicz P.: Podstawy techniki mikroprocesorowej. WNT, Warszawa 1991, rys. 3.34

Zygmunt Kubiak

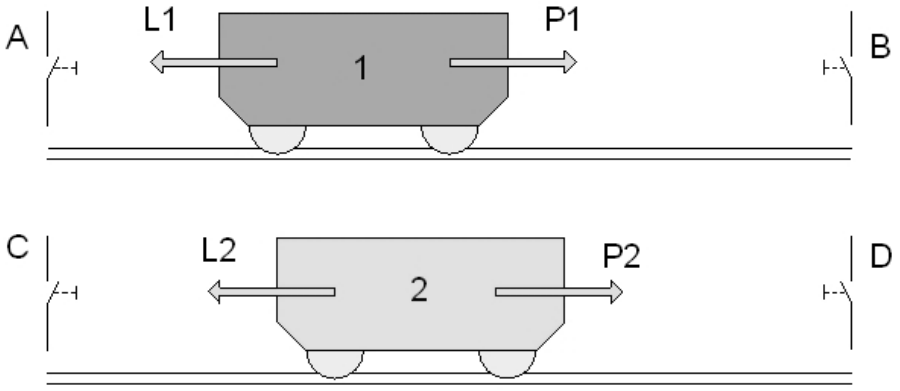
Programowanie systemów wbudowanych (4)

Sterowanie pracą mieszalnika

Cały proces został podzielony na pięć faz. Ponieważ jest to proces sekwencyjny (kolejność następowania poszczególnych etapów w pojedynczym cyklu procesu jest stała) do wskazania fazy aktywnej wykorzystano licznik faz **F**. Na rysunku przedstawiono algorytm synchroniczny, umożliwiający efektywne wykorzystanie czasu mikrokontrolera. Dzięki temu system może realizować współbieżnie wiele innych zadań. Procesor testuje, z maksymalna prędkością, w jakiej fazie znajduje się proces sterowania, wykonuje tylko zadania przewidziane w fazie aktywnej, jeżeli dany krok procesu się zakończył to następuje zwiększenie o 1 licznika faz (w przykładzie modulo 5) i przejście do kolejnego cyklu testowania programu. Taki styl programowania pozwala na łatwa kontrolę poprawności pracy procesora.

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**

Sterowanie zespołu wagoników



L1, P1, L2, P2 - napęd wagoników (lewo, prawo)
A, B, C, D - wyłączniki krańcowe

Sterowanie dwustanowe wagoników

W tym przykładzie przedstawiono bardziej złożony układ sterowania dwustanowego, dotyczący częściowo niezależnego ruchu dwóch wagoników. Algorytm sterowania można opisać następująco:

•w stanie wyjściowym, wagoniki znajdują się w lewym skrajnym położeniu, zwierając wyłączniki krańcowe **A** i **C**;

•naciśnięcie przycisku **S** uruchamia cykl sterowania;

•wagoniki ruszają w prawo (załączenie napędu **P1**, **P2**) – nie jest określone, który dotrze pierwszy do wyłączników krańcowych **B**, **D**;

•wagonik pierwszy, po dotarciu do **B**, wraca natychmiast do **A** (napęd **L1**);

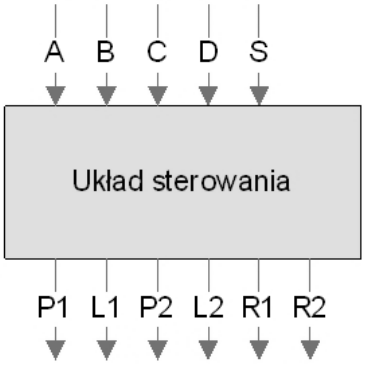
•wagonik drugi, wraca dopiero wówczas (napęd **L2**), gdy pierwszy dotarł do **A**;

•jeżeli przycisk **S** jest nadal włączony, to cykl się powtarza.

Z powyższego opisu wynika, że drugi wagonik będzie czekał w **D** o ile dotarł do **D** przed osiągnięciem **A** przez wagonik pierwszy, w przeciwnym przypadku wraca natychmiast.

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**

Sterowanie zespołu wagoników

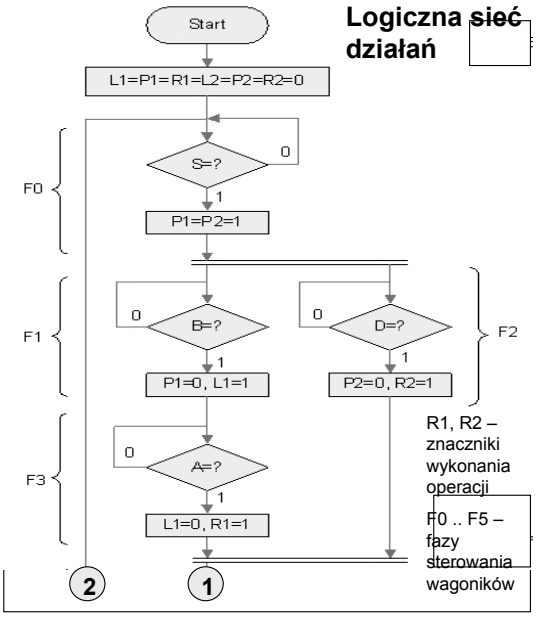


L1, P1, L2, P2 - napęd wagoników (lewo, prawo)
A, B, C, D - wyłączniki krańcowe

Sterowanie dwustanowe wagoników

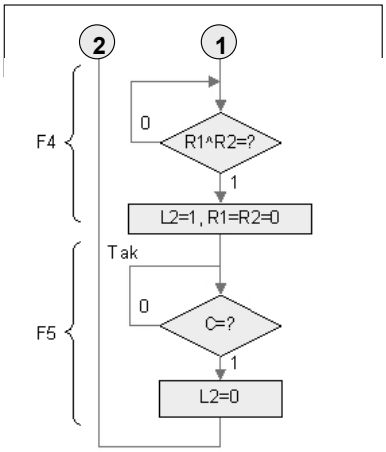
Na wejścia układu sterowania wchodzą sygnały z wyłączników krańcowych (**A, B, C, D**) oraz z przełącznika startu (**S**). Na ich podstawie układ sterowania generuje sygnały wyjściowe sterujące napędami (**L1, P1, L2, P2**). oraz sygnały pomocnicze (nie są konieczne) **R1, R2** wskazujące wykonanie operacji współbieżnych.

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**



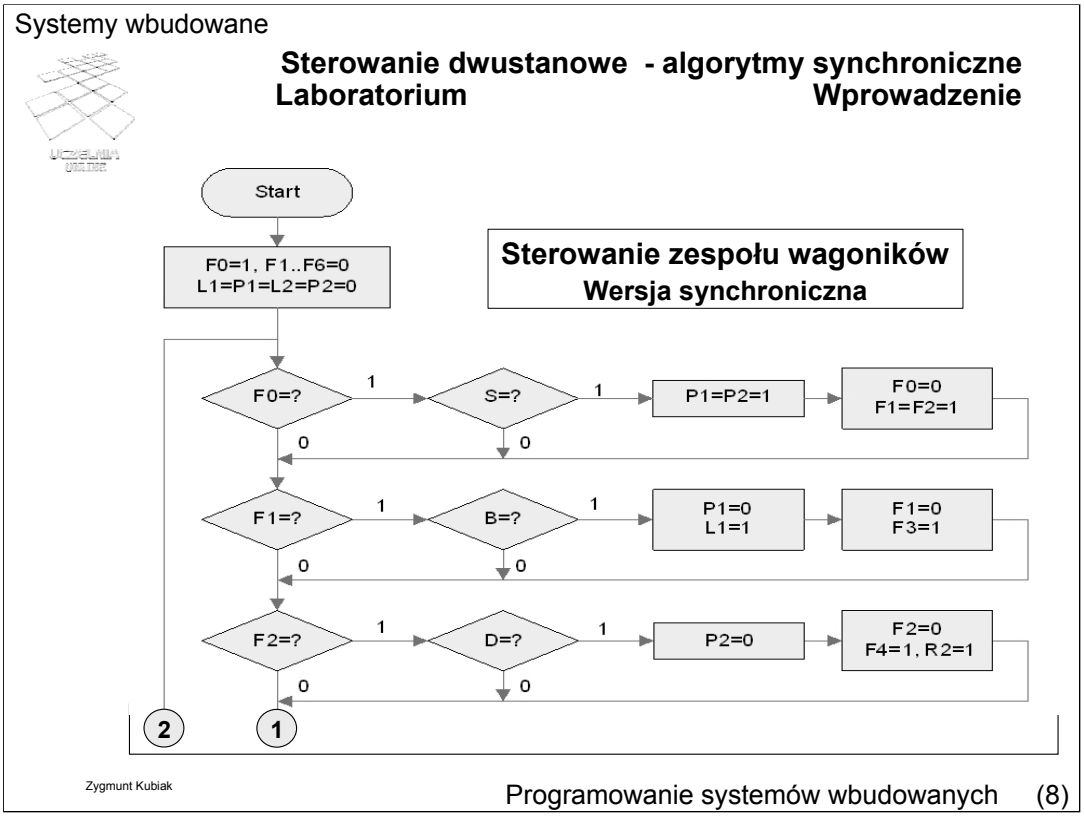
Logiczna sieć działań

Sterowanie zespołu wagoników



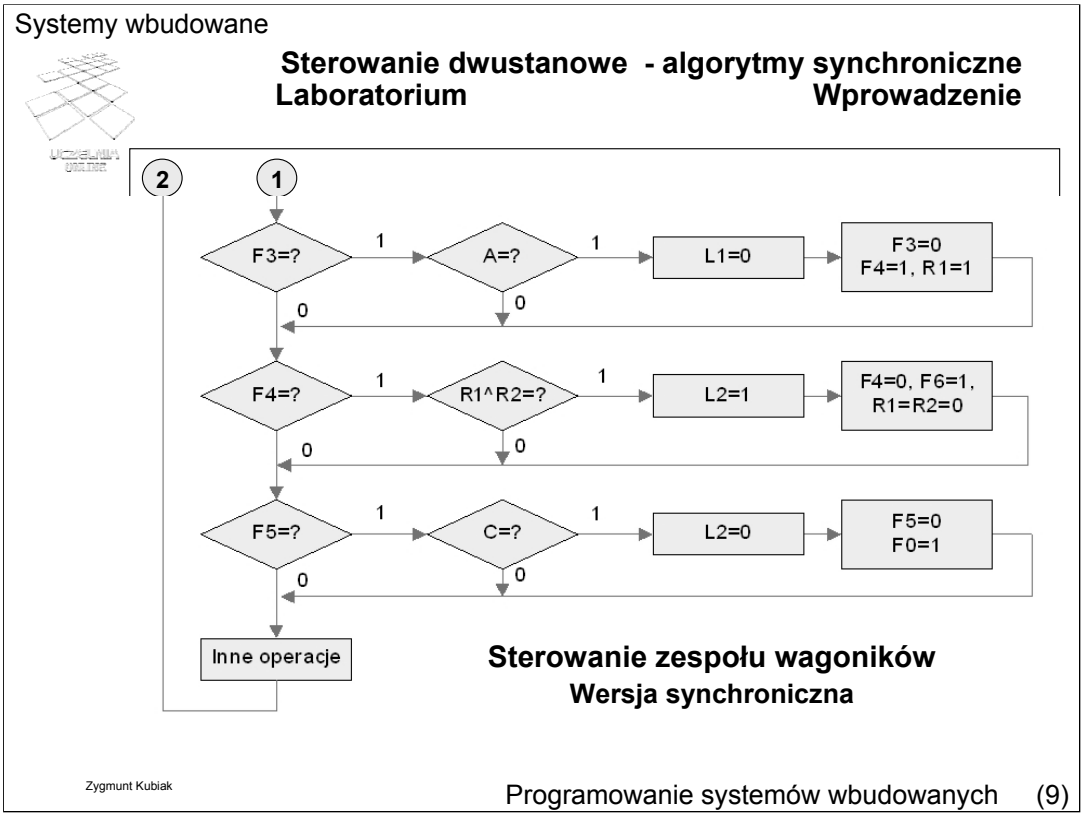
Sterowanie dwustanowe wagoników

Dokładną sieć działań przedstawiono na rysunku. Jak można zauważyć, sieć działań nie jest już prostą pętlą □ zawiera rozgałęzienie r ſwnoległe, kt ſre uwzględnia niezależny ruch obu wagonik ſw. Takiego algorytmu nie można bezpośrednio wykonać na mikrokontrolerze, ponieważ wykonuje on operacje sekwencyjnie. Przekształcenie nawet takiego prostego schematu do postaci czysto sekwencyjnej jest operacją bardzo czasochłonną i dość złożoną. Poza tym tak uzyskany algorytm ma wszystkie wady om ſwione na poprzednim przykładzie.



Sterowanie dwustanowe wagoników

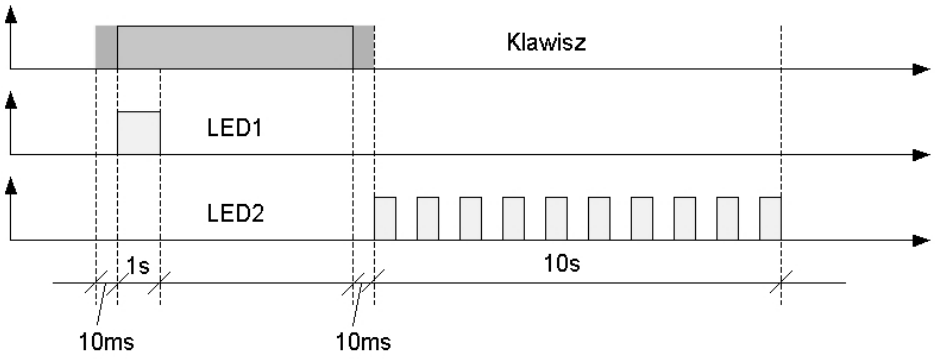
Realizacja synchroniczna jest natomiast w tym przypadku równie prosta jak w przykładzie mieszalnika. Ponieważ w tym przypadku zachodzi jednoczesna aktywność dwóch faz, zamiast licznika faz trzeba zastosować **znaczniki faz**. Ponadto w każdym cyklu skanowania programu, należy przejść do testowania wszystkich faz procesu. Należy również pamiętać o **konieczności synchronizacji wykonania gałęzi równoległych.** Oznacza to, że realizacja połączenia rozgałęzień równoległych wymaga wykonania tych gałęzi przed przejściem do dalszej części programu. W tym celu w algorytmie wprowadzono znaczniki wykonania gałęzi **R1** i **R2**. Wykonanie gałęzi sprawdzane jest przy pomocy warunku iloczynu logicznego: **R1 AND R2**.



Sterowanie dwustanowe wagoników

Należy zauważyć, że tak zbudowany można w prosty sposób rozszerzać przez wprowadzanie kolejnych faz. W przypadku korzystania ze znaczników faz, miejsce wprowadzenia nowej fazy nie ma znaczenia. Dla przedstawionej wersji synchronicznej założono możliwość wystąpienia aktywności więcej niż jednej fazy. Algorytm jest bardzo czytelny – nie wymaga komentarza.

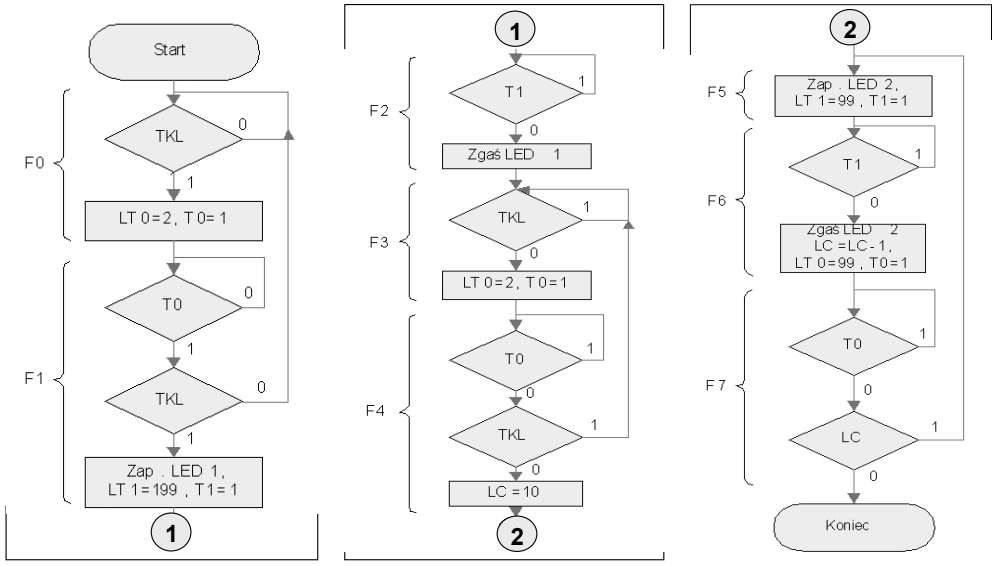
Sterowanie diodami LED



Sterowanie diodami LED

Naciśniecie klawisza powoduje zapalenie LED1 na 1s. Natomiast zwolnienie klawisza uruchamia 10 cykli pulsowania LED2 (okres 1s). Ponadto założono możliwość drgań styków – w celu ich eliminacji, wprowadzono opóźnienie 10ms od wykrycia pierwszego zbocza narastającego (przy naciskaniu klawisza) lub pierwszego zbocza opadającego (przy zwalnianiu klawisza). Poszczególne czasy, występujące w cyklu sterowania, tzn. 10ms, 0,5s i 1s są generowane przy pomocy **timerów programowych T0** i **T1** działających z wykorzystaniem przerwania czasowego 5ms. Program zrealizowany został na podstawie **algorytmu synchronicznego** i dzięki temu w minimalnym stopniu obciąża mikrokontroler.

Sterowanie diodami LED



Sterowanie diodami LED

Przyjęte oznaczenia:

TKL – test stanu klawisza; TX – bit stanu Timera X oraz LTX – licznik (np. typu integer) Timera X; LC – licznik pętli

Algorytm sekwencyjny, przedstawiony na rysunku nie zapewnia wydajnej pracy mikrokontrolera, który większość czasu czeka na naciśnięcie lub zwolnienie klawisza. Aby móc efektywnie wykorzystać czas mikrokontrolera, należy przekształcić algorytm sterowania do wersji synchronicznej.

Aby przejść do wersji synchronicznej należy zaznaczyć na algorytmie sekwencyjnym fazy i powiązać je z odpowiednimi znacznikami (tu: F0 .. F7).

Pętle oczekiwania w programie, związane są z trzema typami zdarzeń:

zdarzenia sprzętowe – naciśnięcie lub zwolnienie klawisza

zdarzenia czasowe – zakończenie odliczania timerów T0 lub T1

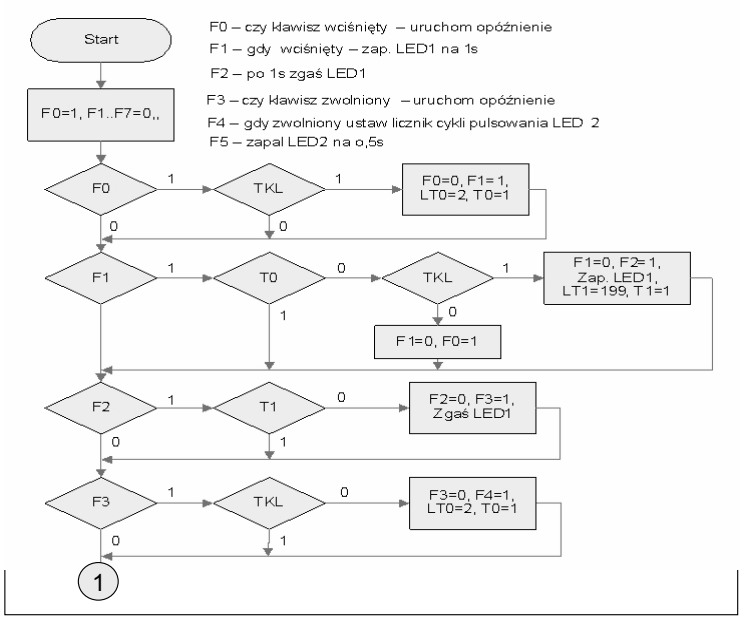
zdarzenia ilościowe – wykonanie określonej liczby pętli – licznik LC.

Programowanie i użycie Timerów programowych przedstawiono w dalszej części.

Systemy wbudowane

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium **Wprowadzenie**

Sterowanie diodami LED



Zygmunt Kubiak

Programowanie systemów wbudowanych (12)

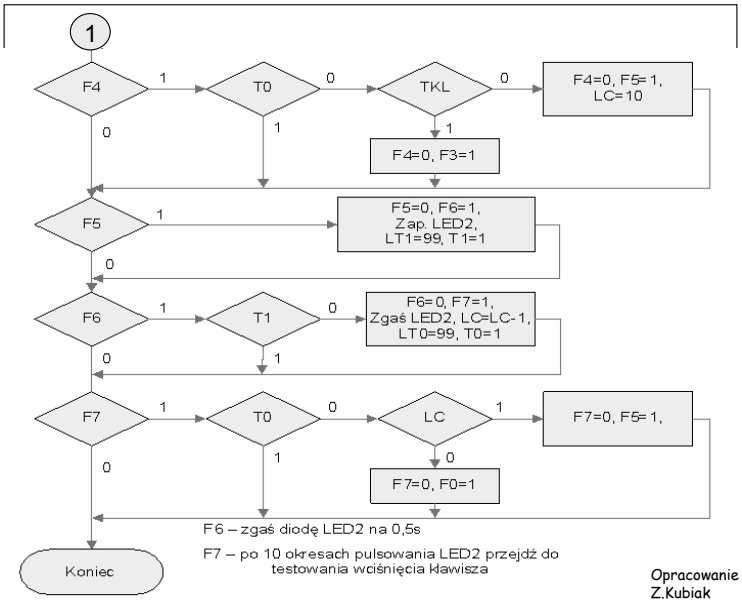
Sterowanie diodami LED

W realizacji synchronicznej mikrokontroler cyklicznie sprawdza, w której fazie znajduje się sterowanie.

W realizacji algorytmów synchronicznych nie ma potrzeby wykorzystywania w programie pętli typu *for .. next.* Faza programu z reguły zawiera blok (-i) decyzji i blok wykonawczy, ale jak pokazano na rysunku, faza może obejmować tylko bloki decyzji (F7) lub tylko blok wykonawczy (F5). Zależy to od programu – blok F5 rozpoczyna pętlę pulsowania diody LED2 i dlatego należało go wyodrębnić.

Sterowanie dwustanowe - algorytmy synchroniczne
Laboratorium Wprowadzenie

Sterowanie diodami LED



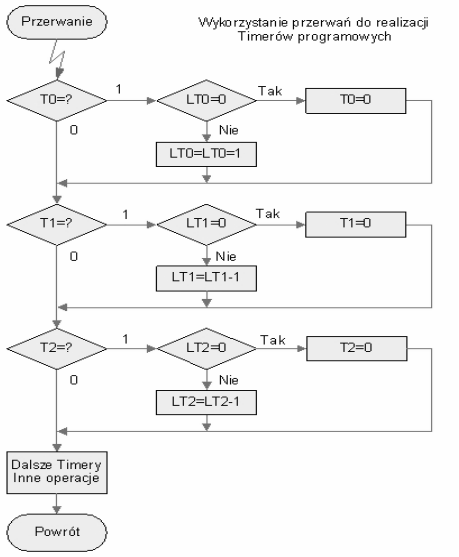
Sterowanie diodami LED

W synchronicznej wersji algorytmu, wykonywane są wyłącznie procedury związane z aktywnymi fazami programu. Program nie czeka na spełnienie warunku. W tej metodzie programowania zaleca się aby fazy były ustawione w kolejności naturalnej, zgodnie z algorytmem. Nie jest to jednak warunek konieczny. Fazy można „przetasować" i nie wpłynie to na działanie programu. Bardzo łatwo rozszerzyć program o nowe fazy. Jest to jedna z podstawowych zalet metody synchronicznej.

Bloki wykonawcze mogą oczywiście zawierać funkcje w j. C. Funkcje, jeżeli zawierają pętle oczekiwania na wystąpienie zdarzenia zewnętrznego lub wewnętrznego, powinny być tworzone również według algorytmu synchronicznego.

Algorytmy synchroniczne stanowią najprostsze rozwiązanie programowania mikrokontrolerów dla systemów czasu rzeczywistego.

Realizacja Timer'ów programowych



Realizacja Timer'ów programowych

W systemach czasu rzeczywistego jednym z problemów jest realizacja zależności czasowych, tzn. opóźnień czasowych, pojedynczych impulsów oraz serii impulsów o zadanym czasie, generatorów przebiegów o zadanej częstotliwości itp. Zwykle żąda się dużej dokładności wytwarzanych przebiegów. Wykorzystywanie do tych celów pętli programowych jest kłopotliwe i mało precyzyjne. Poprawne rozwiązanie tego problemu przedstawione jest poniżej. Proponowane jest wykorzystanie **Timerów programowych**. Mikrokontroler dysponuje niewielką liczbą timerów sprzętowych, np. 2. W programie należy zdefiniować dwa typy zmiennych: **TX** – bit stanu Timera **X** oraz **LTX** – licznik (np. typu integer) Timera **X**. Wartości te ustawiane są w programie głównym, natomiast w programie obsługi przerwań następuje dekrementacja zawartości licznika danego Timera. Po osiągnięciu zera następuje wyzerowanie znacznika Timera, który jest testowany w programie głównym. Użycie Timerów programowych przedstawiono w przykładzie **Sterowanie diodami LED**.

Przedstawiona na rysunku sieć działań pokazuje realizację trzech timerów programowych T0, T1, T2 w podprogramie obsługi przerwań czasowych np. 5ms. Listę timerów można rozszerzać wg potrzeb użytkownika.

Program sterowanie diodami LED

```
//Program był uruchomiony na module Miface1001.
//Schemat i dokumentacja modułu na stronie
//www.aries-rs.com.pl

// Przykład z obsługą Timerów programowych T1, T2 i klawisza
// Naciśniecie klawisza powoduje zapalenie LED1 na 1s
// Zwolnienie klawisza uruchamia 10cykli pulsowania LED2 (okres 1s)
// Autor: Z.Kubiak IIPP, 2006
// Kontroler: ATmega8515
// Naciśnięty klawisz zwiera wejście do masy
// Diody zapalane sygnalem zero

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>

unsigned char LC,T0,T1;
unsigned char F0,F1=0,F2=0,F3=0,F4=0,F5=0,F6=0,F7=0;
unsigned int  LT0,LT1;
```


Sterowanie diodami LED

Realizacja programowa algorytmu synchronicznego, przedstawionego powyżej, dla mikrokontrolera AVR typu Atmega8515.

cd. Program sterowanie diodami LED

```
//Program obsługi przerwania 5ms  (Tck =  11,0592MHz)

SIGNAL  (SIG_OVERFLOW0)
 {
        if (T0==1)
        {   if (LT0==0) T0=0;
                else LT0--;
        }
        if (T1==1)
        {   if (LT1==0) T1=0;
                else LT1--;
        }
        TCNT0 = 202;                    // cykl 54
}
```


Sterowanie diodami LED

Program obsługi przerwań 5ms.

cd. Program sterowanie diodami LED

```
int main(void)
{

//MCUCR = 2;              // przerwania zewnętrzne
DDRD = 0xF0;              // Bity PORTD 0..3 wej; 4..7 wyj
PORTD = 15;               // podciąganie wejść
DDRB = 0xFF;              // PORTB wyjścia
TIMSK = 2;
TCNT0 = 202;
TCCR0 = 5;                // podział przez 1024
F0=1;                     // faza programu
T1=0;                     // znacznik aktywności timera programowego
//F1=0;
//F2=0;
PORTB=255;                // ustaw PORTB - zgaś diody

sei();                    // odblokowanie przerwań
                          // diody LED - PORTB.2 i PORTB.3
```


Sterowanie diodami LED

cd. Program sterowanie diodami LED

```
while(1)
{
        if (F0)
        {   if ((PIND & (1<<3))==0)      //test klawisza czy wcisnięty
                { LT0=2;                 // filtrowanie drgań styków
                  F1=1;
                  F0=0;
                  T0=1;                  // uruchomienie Timera T1 - 10ms
                }
        }
```


Sterowanie diodami LED

cd. Program sterowanie diodami LED

```
if (F1)
{   if (T0==0)
        {   if ((PIND & (1<<3))==0)
                {       PORTB &= ~(1<<2);   // RST bit - zapal diodę LED1
                        LT1=199;
                        F2=1;
                        F1=0;
                        T1=1;               // uruchomienie Timera T2 - 1s
                }
                else
                {       F1=0;
                        F0=1;
                }
        }
}
```


Sterowanie diodami LED

cd. Program sterowanie diodami LED

```
if (F2)
{   if (T1==0)
            {   PORTB |= (1<<2);          // SET bit - wył diodę LED1
            F2=0;
            F3=1;
            }
}
if (F3)
{   if (PIND & (1<<3))                    //test klawisza czy wycisnięty
            {  LT0=2;                     // filtrowanie drgań styków
               F4=1;
               F3=0;
               T0=1;                      // uruchomienie Timera T1 - 10ms
            }
}
```


Sterowanie diodami LED

cd. Program sterowanie diodami LED

```
if (F4)
{   if (T0==0)
            {   if (PIND & (1<<3))
                        {
                                    LC=10;      // 10 cykli pulsowania LED2
                                    F5=1;
                                    F4=0;     }
                        else
                        {           F4=0;
                                    F3=1;     }
            }
}
if (F5)
{           PORTB &= ~(1<<3);               // RST bit - zapal diodę LED2
            LT1=99;
            F5=0;
            F6=1;
            T1=1;
}
```


Sterowanie diodami LED

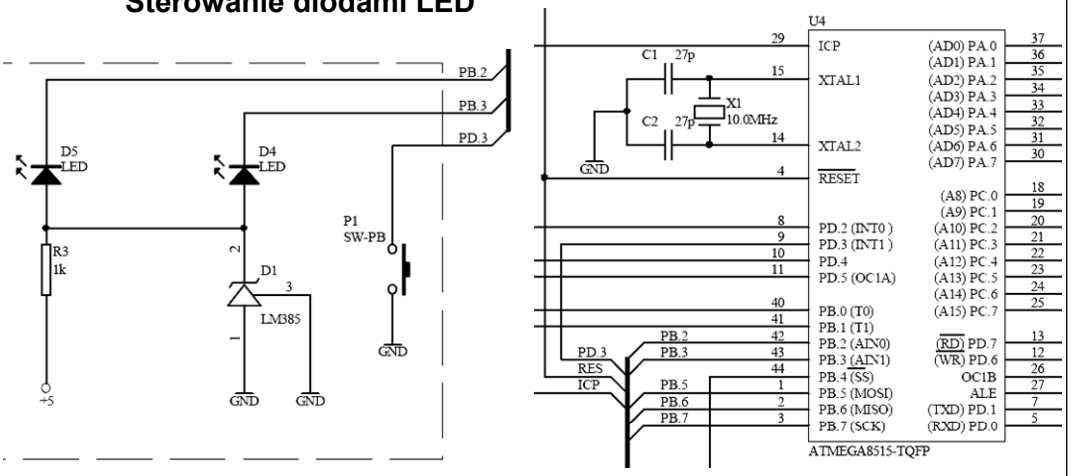
cd. Program testowanie diodami LED

```
if (F6)
{  if (T1==0)
        {  PORTB |= (1<<3);          // SET bit - wył diodę LED2
           LT0=99;
           LC--;
           F6=0;
           F7=1;
           T0=1;      }
}
if (F7)
{       if (T0==0)
        {       if (LC==0)           // czy wszystkie cykle wykonane?
                {       F7=0;
                        F0=1;      }
                else
                {       F7=0;
                        F5=1;      }
        }
}
}
}
```


Sterowanie diodami LED

Również realizacja programowa algorytmów synchronicznych jest bardzo przejrzysta.

Sterowanie diodami LED



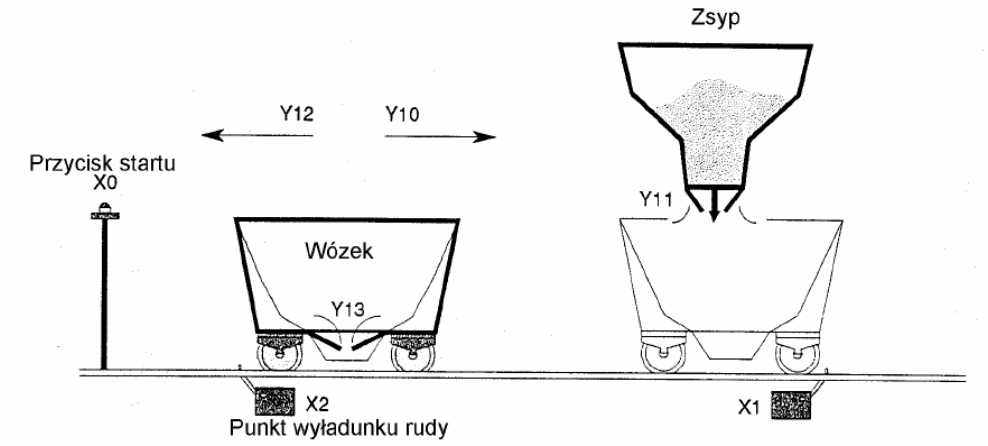
Sterowanie diodami LED

Fragment połączeń sprzętowych modułu dla którego zostało przygotowane oprogramowanie. Jak można zauważyć diody LED zapalane są sygnałem ‘0’ logicznego.

Systemy wbudowane

Laboratorium
Zadanie 1

Zadanie 1 Transport rudy



Zygmunt Kubiak

Programowanie systemów wbudowanych (24)

Zadanie 1

Transporter rudy

Zastosowane oznaczenia: **X0** – przycisk startu, **X1**, **X2** – wyłączniki krańcowe, **Y10** – napęd wózka w prawo, **Y12** – napęd wózka w lewo, **Y11** – napęd ładowania wózka, **Y13** – napęd rozładowania wózka.

Położeniem startowym jest lewe skrajne położenie wózka (**X2=1**) ponadto wózek powinien być rozładowany (**Y13=0** – napęd wyłączony). Po naciśnięciu przycisku startu (**X0=1**), wózek rusza w prawo (**Y10=1** – napęd załączony, **X2=0** – wózek odjechał). Po dotarciu do prawego skrajnego położenia (**X1=1**) wózek się zatrzymuje (**Y10=0**) i rozpoczyna się ładowanie (**Y11=1**), które trwa 8s. Po tym czasie, naładowany wózek rusza w lewo (**Y11=0**, **Y12=1**, **X1=0**). Po dotarciu do punktu wyładunku (**X2=1**) wózek się zatrzymuje (**Y12=0**) i rozpoczyna się rozładowanie wózka (**Y13=1**), które trwa 6s. Po tym czasie **Y13=**0. Jeżeli po upływie czasu rozładowania przycisk startu jest nadal włączony (**X0=**1) to wózek wykonuje kolejny cykl. Jeżeli w międzyczasie został wyłączony (**X0=0**) to wózek pozostaje w tym stanie do kolejnego startu.

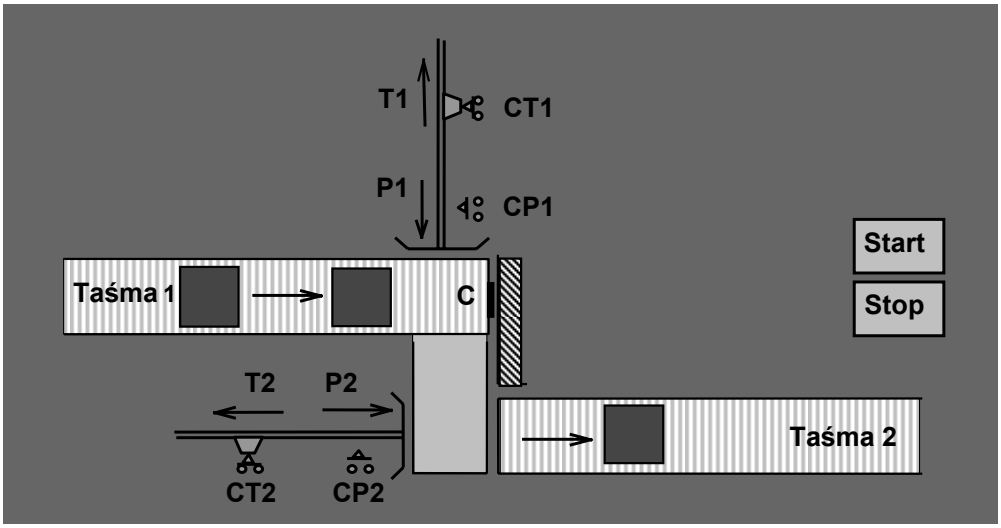
Zadanie 1

1. Narysować sieć działań dla opisanego procesu.
2. Stosując licznik faz lub znaczniki faz przekształcić sieć działań do postaci algorytmu synchronicznego.
3. Opracować program dla wskazanego mikrokontrolera. Uwzględnić programowa realizacje Timerów.

Systemy wbudowane

Laboratorium
Zadanie 2

Zadanie 2 Transporter



Zygmunt Kubiak

Programowanie systemów wbudowanych (25)

TRANSPORTER

Zastosowane oznaczenia: **Start** – przycisk startu; **Stop** – przycisk stop; **C** – czujnik obecności paczki, **CP**, **CT** – Czujniki podajników (przód, tył); **P**, **T** – napędy podajników przód, tył.

Proces uruchamiany jest przyciskiem **Start** a zatrzymywany przyciskiem **Stop**. Paczki jadące na '**Taśmie 1**' muszą zostać przesunięte na '**Taśmę 2**' za pomocą podajników. Podajniki mogą poruszać się w przód i tył (**P,T**) a ich położenia są kontrolowane za pomocą wyłączników krańcowych **CP** i **CT**. Zakłada się, że pudełka są umieszczone w odpowiednio dużych odstępach. Gdy pudełko znajduje się na końcu '**Taśmy 1**' (zamyka się nie zaznaczony na rysunku wyłącznik krańcowy **C**), wówczas podajnik '**1**' przesuwa je pod podajnik '**2**'. Podajnik '**2**' przesuwa pudełko na '**Taśmę 2**', a w tym czasie podajnik '**1**' cofa się. Podajniki po akcji przesuwania wycofują się niezależnie (nie wiadomo, który pierwszy osiągnie położenie **CT**).

(**Wersja uproszczona** – podajnik **1** (górny) wycofuje się po tym jak podajnik **2** przesunął pudełko na **taśmę 2**; podajnik 2 wycofuje się jako drugi, tzn. gdy podajnik 1 osiągnął położenie CT1=1).

Zadanie 2

1. Narysować sieć działań dla opisanego procesu.
2. Stosując licznik faz lub znaczniki faz przekształcić sieć działań do postaci algorytmu synchronicznego.
3. Opracować program dla wskazanego mikrokontrolera.