МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ І СПОРТУ УКРАЇНИ
Національний технічний університет України
“Київський політехнічний інститут”

Коваленко І.О. , Пушкар М.В.

ПРОГРАМНЕ КЕРУВАННЯ ЕЛЕКТРОМЕХАНІЧНИМИ СИСТЕМАМИ -2

Курс лекцій з кредитного модуля для студентів
напряму підготовки 6.050702 "Електромеханіка"
спеціальності "Електромеханічні системи автоматизації та електропривод"
денної форми навчання

Рекомендовано Вченою Радою ФЕА НТУУ «КПІ»

Київ
НТУУ “КПІ”
2013

Програмне керування електромеханічними системами -2: Курс лекцій для студентів напряму підготовки 6.050702 "Електромеханіка" спеціальності "Електромеханічні системи автоматизації та електропривод" денної форми навчання / Уклад. І.О. Коваленко., М.В. Пушкар – К.: НТУУ ”КПІ”, 2013. –117 с.

Гриф надано Вченою радою ФЕА НТУУ "КПІ"
(Протокол № 7 від «25» лютого 2013 р.)

На в ч а л ь н е в и д а н н я

ПРОГРАМНЕ КЕРУВАННЯ ЕЛЕКТРОМЕХАНІЧНИМИ СИСТЕМАМИ -2

Курс лекцій з кредитного модуля для студентів
напряму підготовки 6.050702 "Електромеханіка"
спеціальності "Електромеханічні системи автоматизації та електропривод"
денної форми навчання

Укладачі: Коваленко Ігор Омелянович, канд. техн. наук, доц.
Пушкар Микола Васильович асистент.

Відповідальний редактор М.Я. Островерхов, канд. техн. наук, доц.

Рецензент М.Г. Анпілогов, канд. техн. наук, доц.

ВСТУП

Кредитний модуль «Програмне керування електромеханічними системами-2» призначений для вивчення студентами сучасних мікроконтролерних систем керування двигунами постійного струму та кроковими двигунами.

Метою лекційного курсу є отримання студентами необхідних навиків проектування апаратної частини, розробки алгоритмів та складання керуючих програм на мовах асемблер та С для сучасних систем керування з подальшим застосуванням отриманих навиків при виконанні курсових та дипломних проектів.

У лекційному курсі розглядаються методи побудови та структура сучасних систем керування двигунами постійного струму та кроковими двигунами, виконаних на мікроконтролерній елементній базі.

Особливу увагу приділено програмуванню мікроконтролерів у системах керування.

Для поглибленого вивчення матеріалу подано список рекомендованої літератури.

Лекція 1

ОСНОВНІ ТИПИ ДАНИХ У МОВІ С ТА ПРОГРАМУВАННЯ ЧАСОВИХ ЗАТРИМОК

При розробці керуючих програм завжди прагнуть отримати hex-файли якомога меншого розміру. Цього можна досягти шляхом правильного застосування типів даних мови С. Інакше кажучи, добре розуміння типів даних дозволяє створювати ефективні керуючі програми.

Далі ми розглянемо типи даних, які часто застосовуються в AVR С-компіляторах (табл. 1).

Таблиця 1. Типи даних, що застосовуються С компіляторами

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
char	8-bit	–128 to +127
unsigned int	16-bit	0 to 65,535
int	16-bit	–32,768 to +32,767
unsigned long	32-bit	0 to 4,294,967,295
long	32-bit	–2,147,483,648 to +2,147,483,648
float	32-bit	±1.175e-38 to ±3.402e38
double	32-bit	±1.175e-38 to ±3.402e38

Unsigned char

Тип даних Unsigned char має формат 1 байт і представляє беззнакові числа в діапазоні 0 – 255 (00 - FFH). Це один із типів даних, які найчастіше використовуються в компіляторах AVR. При декларуванні змінних необхідно уважно оцінювати розмір даних і по можливості застосовувати беззнакові числа, замість знакових.

Слід мати на увазі, що С компілятори за замовчуванням встановлюють знаковий тип даних, тому важливо вчасно задекларувати беззнаковий тип за допомогою ключового слова unsigned.

Розглянемо приклади.

Приклад 1

Написати програму, яка виводить в порт B числа від 00 до FF.

Рішення

```
#include <avr/io.h>                          //standard AVR header

int main(void)
{
   unsigned char z;
   DDRB = 0xFF;                              //PORTB is output
   for(z = 0; z <= 255; z++)
      PORTB = z;

   return 0;
}
```

// Зауважимо, що ця програма ніколи не вийде із циклу, оскільки при інкременті

// числа 0xFF результатом буде 0.

Приклад 2

Написати програму, яка виводить в порт B ASCII коди символів 0, 1, 2, 3, 4, 5, A, B, C, D.

Рішення

```
#include <avr/io.h>                          //standard AVR header

int main(void)                               //the code starts from here
{
      unsigned char myList[]= "012345ABCD";
      unsigned char z;
      DDRB = 0xFF;                           //PORTB is output
      for(z=0; z<10; z++)                    //repeat 10 times and increment
            PORTB = myList[z];               //send the character to PORTB

      while(1);                              //needed if running on a traine
      return 0;
}
```

Приклад 3

Написати програму, яка виконує інверсію всіх бітів порту B 200 разів

Рішення

```
//toggle PB 200 times
#include <avr/io.h>                    //standard AVR header

int main(void)                         //the code starts from here
{
      DDRB = 0xFF;                     //PORTB is output
      PORTB = 0xAA;                    //PORTB is 10101010
      unsigned char z;

      for(z=0; z < 200; z++)           //run the next line 200 times
            PORTB = ~ PORTB;           //toggle PORTB

      while(1);                        //stay here forever
      return 0;
}
```

Signed char

Тип даних Signed char має однобітний формат, в якому використовується знаковий біт D7 (із D7 – D0). В результаті, для визначення величини числа залишається 7 бітів, за допомогою яких можна представляти числа від – 128 до +127.

В ситуаціях, коли знак + або – необхідний для представлення конкретної величини (струму, швидкості, температури і т.д.) , вживання знакового типу даних Signed char є обов'язковим, що досягається відповідним декларуванням за допомогою ключового слова Signed char.

Знову нагадаємо, що за замовчуванням у мові С прийнято знаковий тип даних Signed char.

Приклад 4

Написати програму, яка виводить числа від -4 до +4 в порт B

Рішення

```
#include <avr/io.h>                          //standard AVR header

int main(void)
{
   char mynum[ ]  = { -4,-3,-2,-1,0,+1,+2,+3,+4} ;
   unsigned char z;

   DDRB = 0xFF;                              //PORTB is output

   for(z=0; z<=8; z++)
      PORTB = mynum[ z] ;

   while(1);                                 //stay here forever
   return 0;
}
```

// Рекомендується запустити цю програму в симуляторі і подивитися //величини, які виводяться в порт B – 0xFC, 0xFD, 0xFE, 0xFF, 0x00, 0x01, 0x02, //0x03, 0x04 (hex-значення відповідних десяткових чисел -4, -3, -2, -1, 0, 1 і т.д.)

Unsigned int

Цей тип даних має двобайтний формат і може представляти цілі беззнакові числа в діапазоні від 0 до 65535 (0000 - FFFFH).

Знову відзначимо, що за замовчуваням у C компіляторі прийнято знаковий тип даних Signed int.

Signed int

Цей тип даних має двобайтний формат, в якому задіяно знаковий біт D15 (із D15 – D0). Відповідно, ми маємо лише 15 бітів для представлення

величини числа, за допомогою яких можна представляти числа в діапазоні від -32768 до +32767.

Приклад 5

Написати програму, яка виконує інверсію всіх бітів порту B 50000 разів

Рішення

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
   unsigned int z;
   DDRB = 0xFF;                        //PORTB is output

   for(z=0; z<50000; z++)
   {
      PORTB = 0x55;
      PORTB = 0xAA;
   }

   while(1);                           //stay here forever
   return 0;
}
```

// Рекомендується запустити цю програму в симуляторі і подивитися

// як програма виконує інверсію порту B.

// Нагадаємо, що максимальне значення для unsigned int становить 65535.

Приклад 6

Написати програму, яка виконує інверсію всіх бітів порту B 100000 разів

Рішення

```
//toggle PB 100,00 times
#include <avr/io.h>                    //standard AVR header
int main(void)
{
  unsigned long z;                     //long is used because it shou
                                       //store more than 65535.
  DDRB = 0xFF;                         //PORTB is output

  for(z=0; z<100000; z++){
    PORTB = 0x55;
    PORTB = 0xAA;
  }

  while(1);                            //stay here forever
  return 0;
}
```

Програмування часових затримок

Існують три способи створення часових затримок в AVR C, а саме:

- за допомогою простого циклу for;
- шляхом використання попередньо визначених функцій C;
- за допомогою вбудованих апаратних таймерів мікроконтролера.

При програмуванні часових інтервалів за допомогою циклу for необхідно враховувати наступні два фактори:

1. Основним фактором, що визначає час виконання команди і циклу, є частота резонатора, підключеного до зовнішніх виводів XTAL1 – XTAL2.
2. Другим фактором, що впливає на час затримки, є тип компілятора C, який ми застосовуємо. Коли ми програмуємо в асемблері, ми маємо повний контроль над кожною командою та їхньою послідовністю у підпрограмі часової затримки. У випадку C програм компілятор транслює програму на мові C у програму на мові асемблер. В результаті, різні компілятори продукують різні асемблерні коди. Іншими словами, якщо ми компілюємо конкретну програму на C за допомогою різних компіляторів, то отримаємо різні hex-коди.

Зауважимо, що більшість компіляторів виконують оптимізацію програми перед генеруванням hex-коду. В цьому процесі компілятор може взвгалі видалити цикл часової затримки, оскільки в ньому не виконується ніяких змістовних дій, крім затрат машинного часу. Тому рекомендується в таких завданнях встановлювати нульовий рівень оптимізації (оптимізація вимкнута).

Розглянемо приклад.

Приклад 7

Написати програму, яка виконує інверсію всіх бітів порту B із затримкою
100 мс. Мікроконтролер ATmega32, XTAL = 8МГц.

Рішення

```
#include <avr/io.h>                    //standard AVR header
void delay100ms(void)
{
   unsigned int i;
   for(i=0; i<42150; i++);             //try different numbers on your
}                                      //compiler and examine the result.

int main(void)
{
   DDRB = 0xFF;                        //PORTB is output
   while (1)
   {
      PORTB = 0xAA;
      delay100ms();
      PORTB = 0x55;
      delay100ms();
   }
   return 0;
}
```

Інший спосіб формування затримки часу полягає у використанні попередньо визначених функцій C, таких як as_delay_ms() та as_delay_us(). Єдиним недоліком такого підходу є проблема портування програм.

Прииклад 8

Написати програму, яка виконує інверсію всіх бітів порту B із затримкою

10 мс. Застосувати попередньо визначені функції часової затримки у Win AVR.

Рішення

```
#include <util/delay.h>                 //delay loop functions
#include <avr/io.h>                     //standard AVR header

int main(void)
{
       void delay_ms(int d)             //delay in d microseconds
      {
            _delay_ms(d);
      }
      DDRB = 0xFF;                      //PORTA is output
      while (1){
            PORTB = 0xFF;
            delay_ms(10);
            PORTB = 0x55;
            delay_ms(10);
      }
      return 0;
}
```

Контрольні питання

1. Який тип даних встановлюють компілятори С за замовчуванням?
2. Який рівень оптимізації слід встановлювати при організації затримки часу за допомогою циклу?
3. Поясніть способи створення часових затримок.
4. Наведіть максимальні значення чисел для різних типів даних.
5. Поясніть діапазон чисел для типу даних signed char.

Лекція 2

ПРОГРАМУВАННЯ ОСНОВНИХ ОПЕРАЦІЙ НА С

Програмування операцій вводу-виводу

Всі регістри портів мікроконтролерів AVR є доступними у байтовому та бітовому форматах. Далі розглянемо програмування портів у обох форматах.

Обмін інформацією з портами у байтовому форматі.

Для доступу до регістрів порту у режимі виводу ми застосовуємо ім'я PORTx, де x вказує номер порту. У такий же спосіб ми виконуємо операції з регістрами напрямку передачі інформації DDRx.

В режимі вводу ми застосовуємо ім'я порту PINx.

Розглянемо приклади.

Приклад 1
До виводів порту B підключено 8 світлодіодів. Написати програму на С, яка виводить на лінійку світлодіодів двійкові числа від 00 до FFH (від 000 000 до 1111 1111).
Рішення

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
   DDRB = 0xFF;                        //Port B is output
   while (1)
   {
      PORTB = PORTB + 1;
   }
   return 0;
}
```

Приклад 2
До виводів порту B підключено 8 світлодіодів. Написати програму на С, яка вводить байт даних з порту B і виводить його в порт С.
Рішення

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
   unsigned char temp;

   DDRB = 0x00;                        //Port B is input
   DDRC = 0xFF;                        //Port C is output

   while(1)
   {
      temp = PINB;
      PORTC = temp;
   }
   return 0;
}
```

Приклад 3

Скласти програму на С, яка вводить байт даних з порту С. Якщо значення байту менше 100, він виводиться у порт В, інакше – у порт D.

Рішення

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
   DDRC = 0;                           //Port C is input
   DDRB = 0xFF;                        //Port B is output
   DDRD = 0xFF;                        //Port D is output
   unsigned char temp;
   while(1)
   {
      temp = PINC;                     //read from PINB
      if ( temp < 100 )
        PORTB = temp;
      else
        PORTD = temp;
   }
   return 0;
}
```

Доступ до окремих ліній портів

Архітектура мікроконтролерів AVR дозволяє обмін інформацією з окремими розрядами портів (доступ у бітовому форматі). На жаль, не всі AVR С-компілятори підтримують такий режим, а деякі підтримують, але в нестандартний спосіб, що утруднює портування програм.

Наприклад, щоб встановити 1 в нульовому розряді порту B в компіляторі CodeVision необхідно записати

PORTB.0 = 1;

Однак, такий запис не підтримується іншими компіляторами, і WinAVR у тому числі.

Отже, щоб створювати портований код, який може бути відкомпільований в різних компіляторах, необхідно застосовувати логічні оператори AND та OR для доступу до окремих бітів у регістрах.

Таким чином можна отримувати доступ до потрібних бітів без зміни значень інших бітів байту.

Логічні операції в С

Однією із найбільш важливих і потужних особливостей мови С є можливість ефективного виконання бітових операцій.

Всі С- програмісти добре знайомі з логічними операторами AND (&&), OR (||) та NOT (!). Проте, багато С- програмістів в меншій мірі знайомі з побітовими логічними операторами AND (&), OR (|), EX-OR (^), інверсія (~), зсув вправо (>>) та зсув вліво (<<).

Ці побітові оператори широко застосовуються в програмному забезпеченні систем керування та вбудовуваних систем. Отже, добре розуміння цих операторів та вміння правильно їх застосовувати є критичним фактором при розробці мікроконтролерних систем.

Таблиця 1. Побітові логічні оператори в С

		AND	OR	EX-OR	Inverter
A	B	A&B	A|B	A^B	Y=~B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	
1	1	1	1	0	

Наступні приклади показують застосування побітових логічних операторів:

1. 0x35 & 0x0F = 0x05 /* ANDing */
2. 0x04 | 0x68 = 0x6C /* ORing */
3. 0x54 ^ 0x78 = 0x2C /* XORing */
4. ~0x55 = 0xAA /* Inverting 55H */

Розглянемо наступні приклади.

Приклад 4

Розгляньте наведену програму і виконайте аналіз отриманих результатів

Програма

```
#include <avr/io.h>              //standard AVR header
int main(void)
{
   DDRB = 0xFF;                  //make Port B output
   DDRC = 0xFF;                  //make Port C output
   DDRD = 0xFF;                  //make Port D output
   PORTB = 0x35 & 0x0F;          //ANDing
   PORTC = 0x04 | 0x68;          //ORing
   PORTD = 0x54 ^ 0x78;          //XORing
   PORTB = ~0x55;                //inverting
   while (1);
   return 0;
}
```

Приклад 5

Складіть програму на С, яка інвертує лише біт 4 порту В в циклі, не змінюючи значень інших бітів порту

Рішення

```
#include <avr/io.h>                    //standard AVR header

int main(void)
{
   DDRB = 0xFF;                        //PORTB is output

   while(1)
   {
      PORTB = PORTB | 0b00010000;      //set bit 4 (5th bit) of PORTB
      PORTB = PORTB & 0b11101111;      //clear bit 4 (5th bit) of POR
   }

   return 0;
}
```

Приклад 6

Складіть програму на С, яка виконує перевірку біту 5 порту С. Якщо цей біт = 1, вивести значення 0x55 в порт В, інакше вивести значення 0xAA

Рішення

```
#include <avr/io.h>                //standard AVR header

int main(void)
{
   DDRB = 0xFF;                    //PORTB is output
   DDRC = 0x00;                    //PORTC is input
   DDRD = 0xFF;                    //PORTB is output

   while(1)
   {
      if (PINC & 0b00100000)       //check bit 5 (6th bit) of PI
        PORTB = 0x55;
      else
        PORTB = 0xAA;
   }

   return 0;
}
```

Приклад 7

Датчик стану дверей підключено до лінії 1 порту В. Світлодіод підключено до лінії 7 порту С. Скласти програму, яка перевіряє стан дверей і вмикає світлодіод, коли двері відчинені.

Рішення

```
#include <avr/io.h>                    //standard AVR header

int main(void)
{
   DDRB = DDRB & 0b11111101;           //pin 1 of Port B is input
   DDRC = DDRC | 0b10000000;           //pin 7 of Port C is output

   while(1)
   {
     if (PINB & 0b00000010)            //check pin 1 (2nd pin) of PIN
       PORTC = PORTC | 0b10000000;     //set pin 7 (8th pin) of PORTC
     else
       PORTC = PORTC & 0b01111111;     //clear pin 7 (8th pin) of POR
   }
   return 0;
}
```

Контрольні питання

1. Поясніть операцію 0x35&0x0F та вкажіть її результат.
2. Поясніть операцію 0x04|0x68 та вкажіть її результат.
3. Поясніть, яким чином виконується доступ до окремих бітів байту.
4. Поясніть операцію ~0x55 та вкажіть її результат.
5. Поясніть операцію 0x54^0x78 та вкажіть її результат.

Лекція 3

ПЕРЕТВОРЕННЯ ДАНИХ НА МОВІ С

Багато мікроконтролерів мають вбудований годинник реального часу, який визначає час та дату навіть при відключеному основному живленні. Дуже часто дані від годинника реального часу представлені у двійково-десятковому коді. Далі розглянемо, яким чином провадиться перетворення різних форматів даних на мові С.

Представлення цифрової інформації у коді ASCII

Стандартна комп’ютерна клавіатура виконує кодування інформації у коді ASCII. Наприклад, коли натискається клавіша «0», у комп'ютер передається код «0011 0000». Аналогічно, при натисненні клавіші «1» генерується код «0011 0001» і т.д.

У табл. 1 наведено кодування десяткових цифр у коді ASCII, двійковому коді (Binary) та двійково-десятковому коді (BCD).

Таблиця 1. Кодування цифр 0 – 9

Key	ASCII (hex)	Binary	BCD (unpacked)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

Перетворення упакованого двійково-десяткового коду (BCD) у код ASCII

Вбудований годинник реального часу видає інформацію про поточний час безперервно, навіть при вимкненому основному живленні мікроконтролера. Ця інформація представлена в упакованому двійково-десятковому коді (BCD) . Для преретворення а код ASCII її необхідно спочатку розпакувати. Потім на розпаковане двійково-десяткове число накладається маска 0011 0000 (30H).

Цей процес показано на прикладах:

```
Packed BCD     Unpacked BCD          ASCII
0x29           0x02, 0x09            0x32, 0x39
00101001       00000010,00001001     00110010,00111001
```

Перетворення коду ASCII в упакований двійково-десятковий код (BCD)

Для перетворення коду ASCII в упакований двійково-десятковий код необхідно спочатку розпакувати двійково-десятковий код (звільнитися від 3), і потім виконати комбінацію цифр для отримання упакованого двійково-десяткового коду.

Наприклад, при натисканні клавіш «4» та «7» на клавіатурі отримуємо коди 34H та 37H, відповідно. Метою перетворення є отримання коду 47H, або «0100 0111», що є упакованим кодом.

Наприклад:

Key	ASCII	Unpacked BCD	Packed BCD
4	34	00000100	
7	37	00000111	01000111 or 47H

Розглянемо приклади.

Приклад 1
Написати програму на С, яка виконує перетворення упакованого коду (BCD) 0x229 в код ASCII і виводить байти в порти B та C
Рішення

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
   unsigned char x, y;
   unsigned char mybyte = 0x29;

   DDRB = DDRC = 0xFF;                 //make Ports B and C outpu
   x = mybyte & 0x0F;                  //mask upper 4 bits
   PORTB = x | 0x30;                   //make it ASCII
   y = mybyte & 0xF0;                  //mask lower 4 bits
   y = y >> 4;                         //shift it to lower 4 bits
   PORTC = y | 0x30;                   //make it ASCII

   return 0;
}
```

Приклад 2

Написати програму на С, яка виконує перетворення кодів ASCII цифр «4» та «7» в упакований код (BCD) і виводить його в порт В

Рішення

```
#include <avr/io.h>              //standard AVR header

int main(void)
{
   unsigned char bcdbyte;
   unsigned char w = '4';
   unsigned char z = '7';
   DDRB = 0xFF;                  //make Port B an output
   w = w & 0x0F;                 //mask 3
   w = w << 4;                   //shift left to make upper BCD digit
   z = z & 0x0F;                 //mask 3
   bcdbyte = w | z;              //combine to make packed BCD
   PORTB = bcdbyte;

   return 0;
}
```

Перетворення двійкового коду в десятковий і код ASCII

Таке перетворення дуже часто застосовується в системах з мікроконтролерами. Наприклад, вбудований АЦП мікроконтролера видає

результат у двійковому коді. Існують також годинники реального часу, які видають дані у двійковому коді. Для перетворення двійкових даних у код ASCII, їх необхідно спочатку перетворити у двійково-десятковий код, а потім в код ASCII.

Оскільки шістнадцятиричний формат є стандартним способом представлення двійкового коду, ми будемо писати двійкові числа у HEX-форматі. Двійкові числа 00 – FFH, конвертовані у десяткові, дають нам ряд чисел 000 – 255. Один із способів виконати таке перетворення полягає в діленні на 10 числа і залишків.

Наприклад, двійкове число 1111 1101 (FDH) у десятковому виді є 253. Одна із версій алгоритму перетворення виглядає так:

Hex	Quotient	Remainder
FD/0A	19	3 (low digit) LSD
19/0A	2	5 (middle digit)
		2 (high digit) (MSD)

Нижче наведено програмну реалізацію цього алгоритму.

Приклад 3
Написати програму на C, яка конвертує двійкове число 11111101 (FDH) в десяткове і виводить його в порти B, C та D
Рішення

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
     unsigned char x, binbyte, d1, d2, d3;
     DDRB = DDRC = DDRD =0xFF;         //Ports B, C, and D output
     binbyte = 0xFD;                   //binary (hex) byte
     x = binbyte / 10;                 //divide by 10
     d1 = binbyte % 10;                //find remainder (LSD)
     d2 = x % 10;                      //middle digit
     d3 = x / 10;                      //most-significant digit (MSD)
     PORTB = d1;
     PORTC = d2;
     PORTD = d3;

     return 0;
}
```

Контрольні питання

1. Поясніть необхідність перетворення кодів даних.
2. Поясніть представлення цифрової інформації у коді ASCII.
3. Як виконується перетворення десяткового числа у двійково-десяткове?
4. Поясніть алгоритм перетворення перетворення двійкових даних у код ASCII.
5. Як виконується перетворення коду ASCII в упакований двійково-десятковий код (BCD)

Лекція 4

ПРИНЦИПИ ПОБУДОВИ СИСТЕМ КЕРУВАННЯ ДВИГУНАМИ ПОСТІЙНОГО СТРУМУ

Двигуни постійного струму (ДПС) широко застосовуються в різних сферах народного господарства. Отже, створення ефективних систем керування такими двигунами є актуальною задачею. Далі будемо розглядати в якості об'єкта керування двигуни постійного струму з незалежним збудженням. Як відомо, швидкість такого двигуна визначається прикладеною до зажимів якоря напругою, а момент – величиною струму якоря.

На рис. 1 показані схеми, за допомогою яких можна змінювати напрям обертання валу двигуна постійного струму зі збудженням від постійних магнітів. Очевидно, що для зміни напряму обертання валу необхідно міняти полярність напруги, підведеної до якоря двигуна.

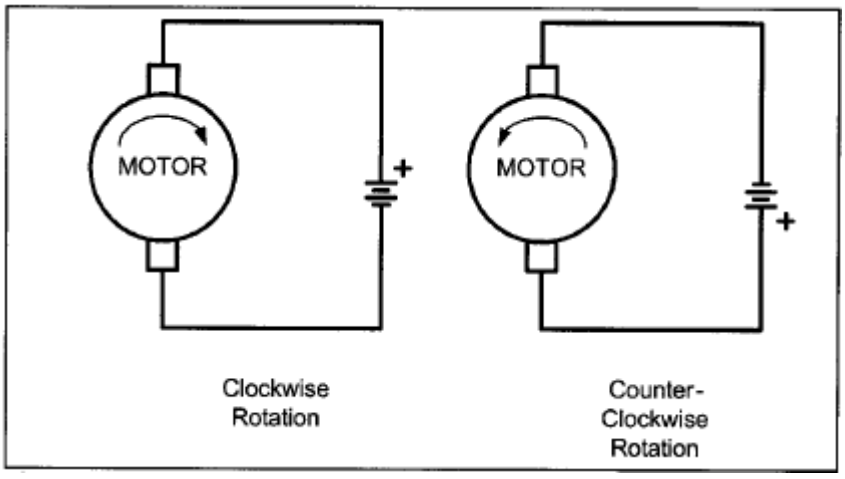


Рис.1- Зміна напряму обертання валу ДПС

На практиці для зміни полярності напруги на якорі двигуна застосовують мостову схему типу Н, як показано на рис. 2. Ця схема містить 4 силових ключі, в якості яких можуть застосовуватися контакти реле, MOS-транзистори або IGBT-транзистори.

Мостова схема має три робочих стани:

- всі ключі розімкнуті – двигун вимкнено;
- замкнуті ключі 1 – 4 – умовний режим роботи «Вперед»;
- замкнуті ключі 2 – 3 –умовний режим роботи «Назад».

Наведена мостова схема має також три аварійних стани:

- замкнені ключі 1 – 3;
- замкнені ключі 2 – 4;
- замкнені ключі 1 – 2 – 3 – 4.

Очевидно, що будь-який із наведених аварійних станів приводить до короткого замикання джерела живлення, супроводжується аварійним відключенням і, можливо, пошкодженням елементів силового кола. Зазвичай в аварійних режимах в першу чергу виходять з ладу силові ключі, як елементи з найменшою термічною стійкістю. Схема аварійних режимів представлена на рис. 3.

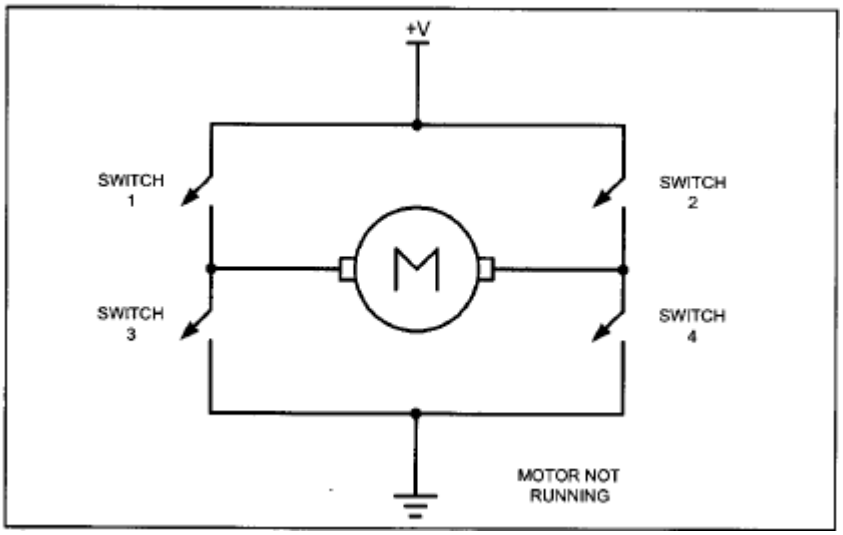


Рис. 2- Мостова схема

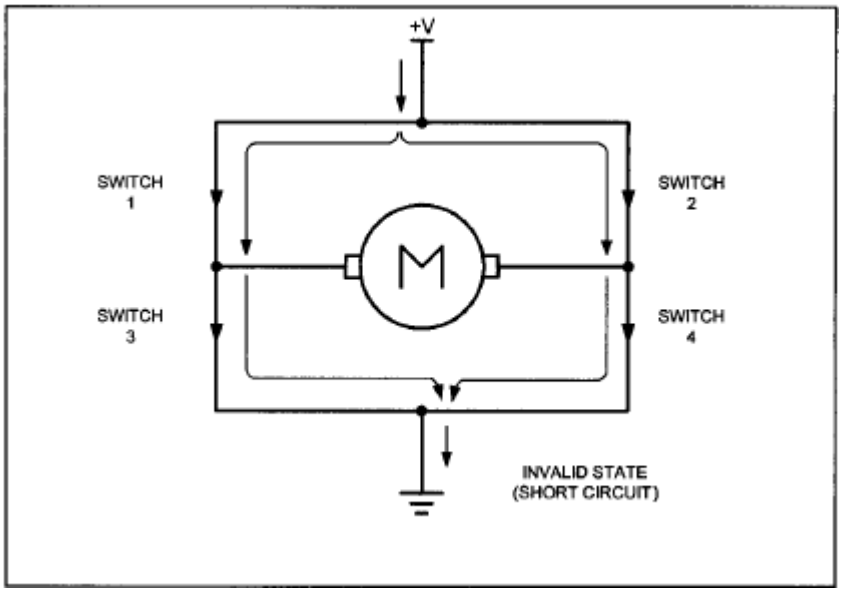


Рис. 3 – Схема аварійних режимів

Приклад 1

Контакти вимикача підключено до лінії порту PINA.7. Силові ключі SW1, SW2, SW3, SW4 підключені відповідно до ліній порту PORTB.1, PORTB.2, PORTB.3, PORTB.4.

Необхідно реалізувати такий алгоритм роботи:

- якщо PA7=0, то режим роботи двигуна «Вперед»;
- якщо PA7=1, то режим роботи двигуна «Назад».

Програма

```
.INCLUDE "M32DEF.INC"
            SBI   DDRB,0          ;make PB0 an output (switch1)
            SBI   DDRB,1          ;make PB1 an output (switch2)
            SBI   DDRB,2          ;make PB2 an output (switch3)
            SBI   DDRB,3          ;make PB3 an output (switch4)
            CBI   DDRA,7          ;make PA7 an input
MONITOR:    SBIS  PINA,7          ;skip next if PINA.7 is set
            RJMP  CLKWISE         ;if PA7 = 0 go to CLKWISE
            CBI   DDRB,1          ;switch2 = 0
            CBI   DDRB,2          ;switch3 = 0
            SBI   DDRB,0          ;switch1 = 1
            SBI   DDRB,3          ;switch4 = 1
            JMP   MONITOR
CLKWISE:    CBI   DDRB,0          ;switch1 = 0
            CBI   DDRB,3          ;switch4 = 0
            SBI   DDRB,1          ;switch2 = 1
            SBI   DDRB,2          ;switch3 = 1
            JMP   MONITOR
```

Спеціальний інтерфейс для керуванням ДПС

Енергетичний рівень сигналів на лініях портів МК зазвичай недостатній для безпосереднього керування ДПС. Тому застосовують спеціалізований інтерфейс, який містить у своєму складі:

- пристрій оптоізоляції (оптоізолятор);
- мостову схему силових ключів за необхідності реверсивної роботи, або окремі силові ключі, якщо реверс непотрібен.

На рис. 4 представлено схему із застосуванням оптоізолятора ILQ74 та силової інтегральної мікросхеми L298N, яка містить силову мостову схему на біполярних транзисторах.

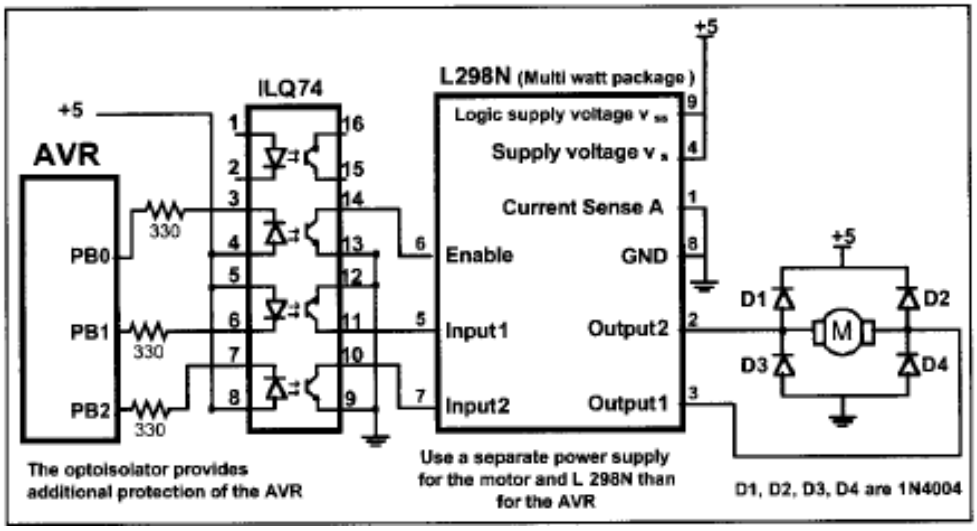


Рис. 4 – Реверсивна схема керування ДПС

Застосування в інтерфейсі оптоізолятора захищає МК високої напруги живлення двигуна, а також від імпульсних завад з боку колекторного вузла двигуна. Отже, враховуючи низьку вартість оптоізолятора, його застосування можна вважати бажаним у більшості випадків.

Приклад 2

На рис. 4 показано підключення мікросхеми L298 до мікроконтролера. Умовно будемо вважати, що до лінії порту PA.7 (PORTA.7) підключено вимикач SW, який дозволяє задавати рівень логічного сигналу на цій лінії. Необхідно реалізувати такий алгоритм роботи:

- якщо SW = 0, то режим роботи двигуна «Вперед»;
- якщо SW = 1, то режим роботи двигуна «Назад».

Програма

```
.INCLUDE "M32DEF.INC"
            SBI   DDRB,0          ;make PB0 an output (Enable)
            SBI   DDRB,1          ;make PB1 an output (clock)
            SBI   DDRB,2          ;make PB2 an output (counter)
            SBI   PORTB,0         ;Enable = 1
            CBI   DDRA,7          ;make PA7 an input
            SBI   PORTA,7
MONITOR:    SBIS  PINA,7          ;skip next if PINA.7 is set
            RJMP  CLKWISE         ;if PA7 = 0 go to CLKWISE
            CBI   PORTB,1         ;switch1 = 0
            SBI   PORTB,2         ;switch2 = 1
            JMP   MONITOR
CLKWISE:    SBI   PORTB,1         ;switch1 = 0
            CBI   PORTB,2         ;switch2 = 1
            JMP   MONITOR
```

У застосуваннях з нереверсивним режимом роботи схеми приводу можуть бути значно спрощені. На рис. 5 наведено схему приводу з нереверсивним режимом роботи і такі принципові рішення можуть бути застосовані для багатьох інших застосувань. Типовим є використання оптоізолятора для відокремлення інформаційних кіл МК від високої напруги живлення двигуна та завад, що виникають при роботі колекторного вузла.

На рис. 5 показана схема із застосуванням біполярного силового транзистора в якості силового ключа. Застосування оптоізолятора дозволяє:

- подавати на двигун високу напругу без ризика пошкодження МК;
- підвищити надійність роботи системи внаслідок зменшення негативного впливу завад від двигуна на МК.

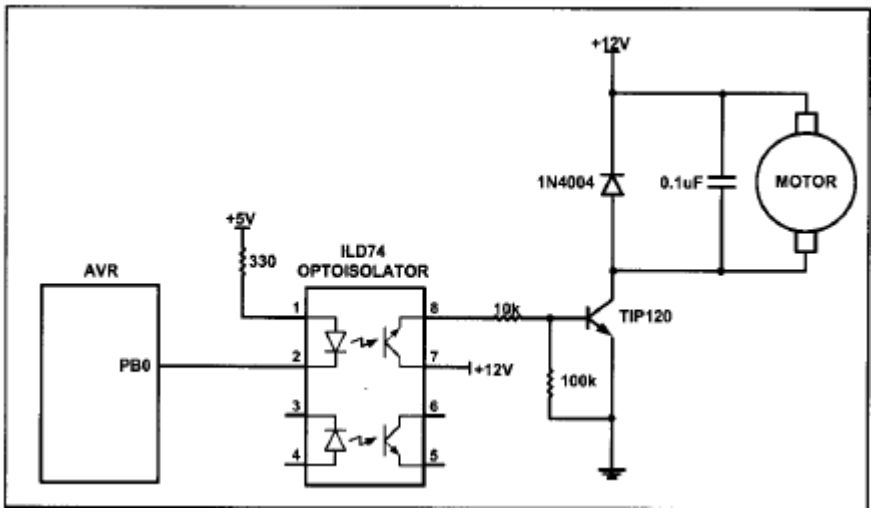


Рис. 5 – Нереверсивний привод з силовим біполярним транзистором

Зауважимо, що силовий ключ вмикається при нульовому сигналі на лінії порту PB0.

Контрольні питання

1. Поясніть, як будується інтерфейс для керуванням ДПС.
2. Які типи силові ключів застосовуються у системах керування двигунами?
3. Чому необхідно застосовувати оптоізолятори?
4. Які аварійні режими можливі в мостовій схемі?
5. Поясніть схемотехніку нереверсивних та реверсивних систем керування двигунами.

Лекція 5

ЗАСТОСУВАННЯ ШИРОТНО-ІМПУЛЬСНОЇ МОДУЛЯЦІЇ

Широтно-імпульсна модуляція (ШІМ — англ. PWM), або модуляція за тривалістю імпульсів (англ. PDM) — процес керування шириною (тривалістю) високочастотних імпульсів по закону, який задає низькочастотний сигнал. В силовій електроніці це може бути керування середнім значенням вихідної напруги шляхом зміни тривалості замкнутого стану електронного (електромеханічного) ключа, наприклад, у схемі ключового стабілізатора напруги.

Перемикання відбувається з великою швидкістю, такою яка потрібна для даного типу навантаження. Це може бути декілька разів на хвилину для електропечі; 100 Гц для електролампи; від декількох до десятків кГц для електродвигуна або від десятків до сотень кГц для аудіопідсилювача і комп'ютерного блоку живлення.

Для оцінки ефективності керування має значення робочий цикл (англ. duty cycle) під яким розуміється відношення тривалості ввімкненого стану (англ. 'on' time) до прийнятого періоду імпульсів ; малий duty cycle відповідає режиму енергозберігання, оскільки джерело енергії відключене більшість часу. Duty cycle виражають в відсотках, 100% відповідає увімкненому стану на весь період.

Головною перевагою ШІМ є мала втрата енергії на електронному перемикачеві. Він здебільшого перебуває або у вимкнутому стані, коли його опір максимальний, або в режимі насичення — з мінімальним опором, тобто або струм, або падіння напруги на ньому близькі до нуля. ШІМ також органічно вписується в цифрові технології оскільки велика кількість ШІМ-контролерів виробляється в вигляді мікросхем. Класичним прикладом є мікросхеми UC3842 ...3844. Також більшість сучасних мікроконтролерів оснащені вбудованими широтно-імпульсними модуляторами.

З точки зору класифікації цифрова широтно-імпульсна модуляція є різновидом дворівневої імпульсно-кодової модуляції (ІКМ).

Швидкість двигуна постійного струму визначається трьома факторами:

- напругою;

- струмом;

- навантаженням на валу двигуна.

Для регулювання середнього значення напруги на якорі двигуна застосовується ШІМ. За допомогою зміни (модуляції) ширини імпульсів напруги, підведеної до якоря двигуна, можливо регулювати кількість енергії, що постачається двигуну і його швидкість. Таким чином, збільшення ширини імпульсу призводить до зростання швидкості двигуна.

Широтно-імпульсна модуляція настільки широко застосовується в системах керування, що багато мікроконтролерів оснащені вбудованими пристроями ШІМ. При застосуванні таких мікроконтролерів практична реалізація системи керування з ШІМ значно спрощується – досить лише задати бажані параметри ШІМ, а всі інші рутинні операції виконає мікроконтролер.

На рис. 1 наведено застосування MOS-транзистора в якості силового ключа у схемі нереверсивного приводу. Стабілітрон (діод Зенера) обмежує відкриваючий сигнал на затворі транзистора на припустимому рівні.

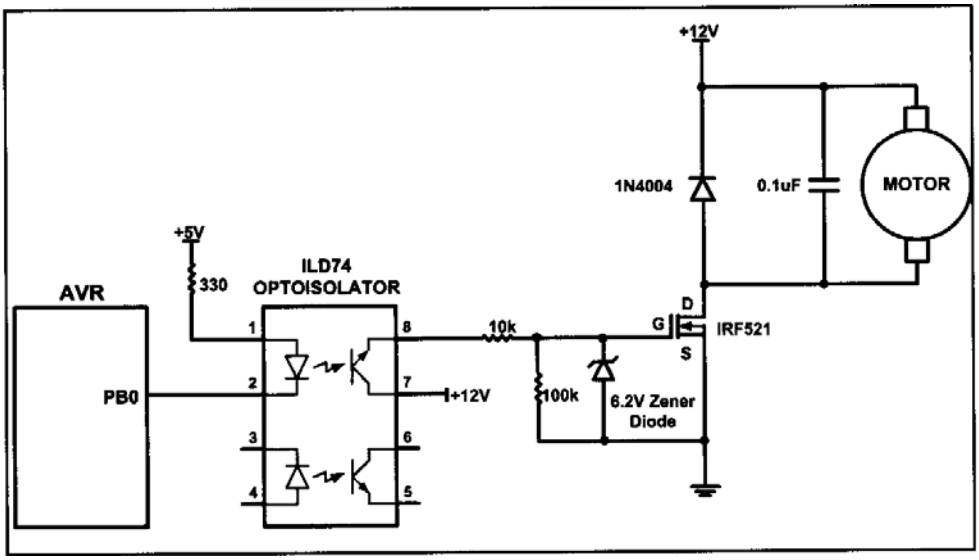


Рис. 1 – Застосування MOS-транзистора в нереверсивному приводі

Часові діаграми сигналів з широтно-імпульсною модуляцією показані на рис. 2. На цих епюрах можна бачити зміну виду сигналу по мірі збільшення тривалості імпульсу, починаючи від 25% до 100%.

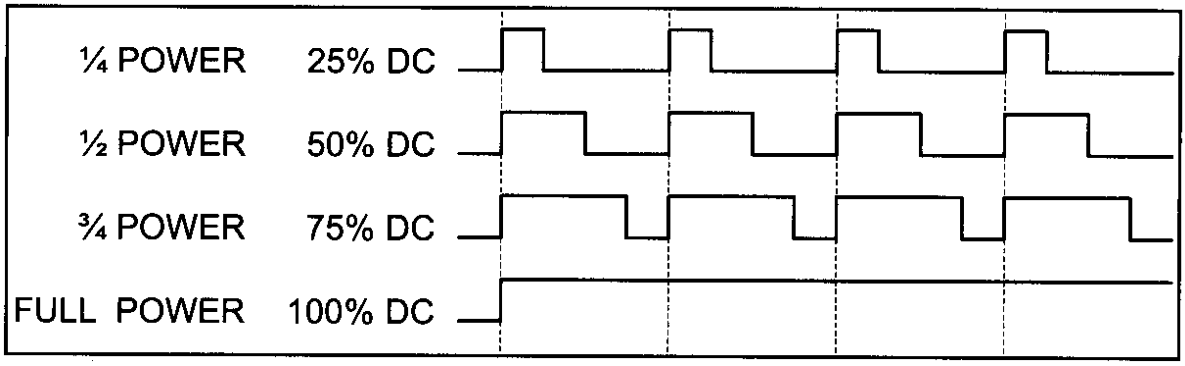


Рис. 2 – Часові діаграми сигналів з ШІМ

Програмна реалізація ШІМ

Розглянемо схему нереверсивного приводу з біполярним транзистором в якості силового ключа. На цій схемі увімкнутому стану ключа відповідає сигнал на лінії порту PORTB.0 = 0.

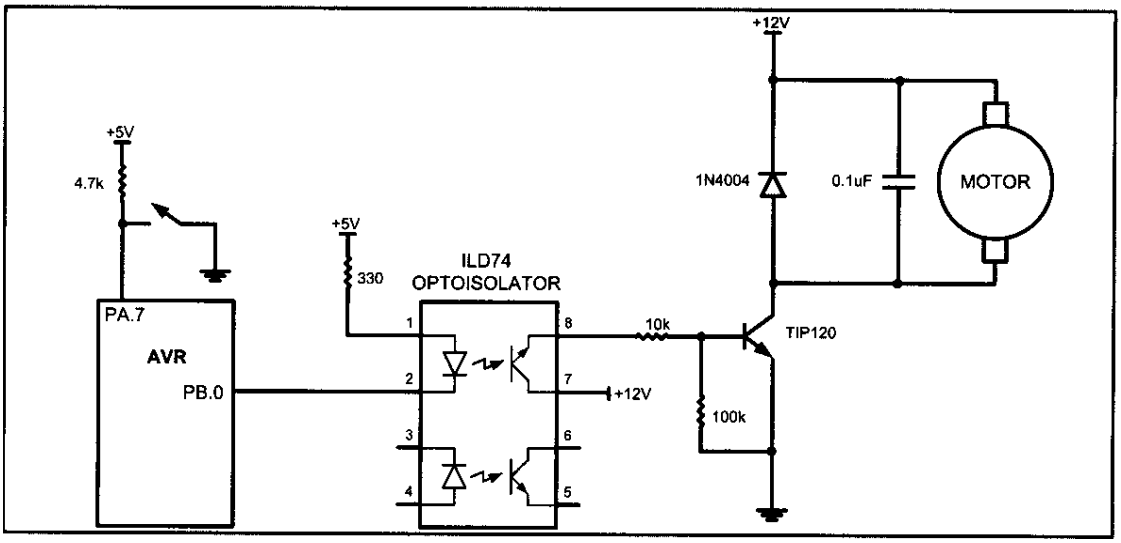


Рис. 3 – Схема нереверсивного приводу

Складемо програму на асемблері, яка генерує на лінії PORTB.0 широтно-імпульсний сигнал:

- з тривалістю імпульсу 25%, якщо PORTA.7 = 1;
- з тривалістю імпульсу 50%, якщо PORTA.7 = 0.

Текст програми

```
.INCLUDE "M32DEF.INC"
      LDI   R16,HIGH(RAMEND)
      OUT   SPH,R16
      LDI   R16,LOW(RAMEND)
      OUT   SPL,R16          ;initialize stack pointer
      SBI   DDRB,0           ;PORTB.0 as output
      CBI   DDRA,7           ;PORTA.7 as input
      SBI   PORTA,7          ;enable pull-up
      CBI   PORTB,0          ;turn off motor
CHK:  SBIC  PINA,7
      RJMP  P50
      SBI   PORTB,0          ;high portion of pulse
      RCALL DELAY
      RCALL DELAY
      RCALL DELAY
      CBI   PORTB,0          ;low portion of pulse
      RCALL DELAY
      RJMP  CHK
P50:  SBI   PORTB,0          ;high portion of pulse
      RCALL DELAY
      RCALL DELAY
      CBI   PORTB,0          ;low portion of pulse
      RCALL DELAY
      RCALL DELAY
      RJMP CHK
```

За незмінних умов перепишемо цю програму на мові С. Отримаємо наступний текст програми.

```
#define F_CPU    8000000UL       //XTAL = 8 MHz
#define SW       (PORTA&(1<<7))

#include "avr/io.h"
#include "util/delay.h"

void main()
{
    DDRA=0x7F;          //make PA7 input pin
    DDRB=0x01;          //make PB0 output pin
    while(1)
    {
        if(SW == 1)
        {
             PORTB = PORTB | (1<<0);
             _delay_ms(75);
             PORTB = PORTB & (~(1<<0));
             _delay_ms(25);
        }
        else
        {
             PORTB = PORTB | (1<<0);
             _delay_ms(50);
             PORTB = PORTB & (~(1<<0));
             _delay_ms(50);
        }
     }
}
```

Далі розглянемо програму для реверсивної схеми приводу, наведеної на рис. 4.

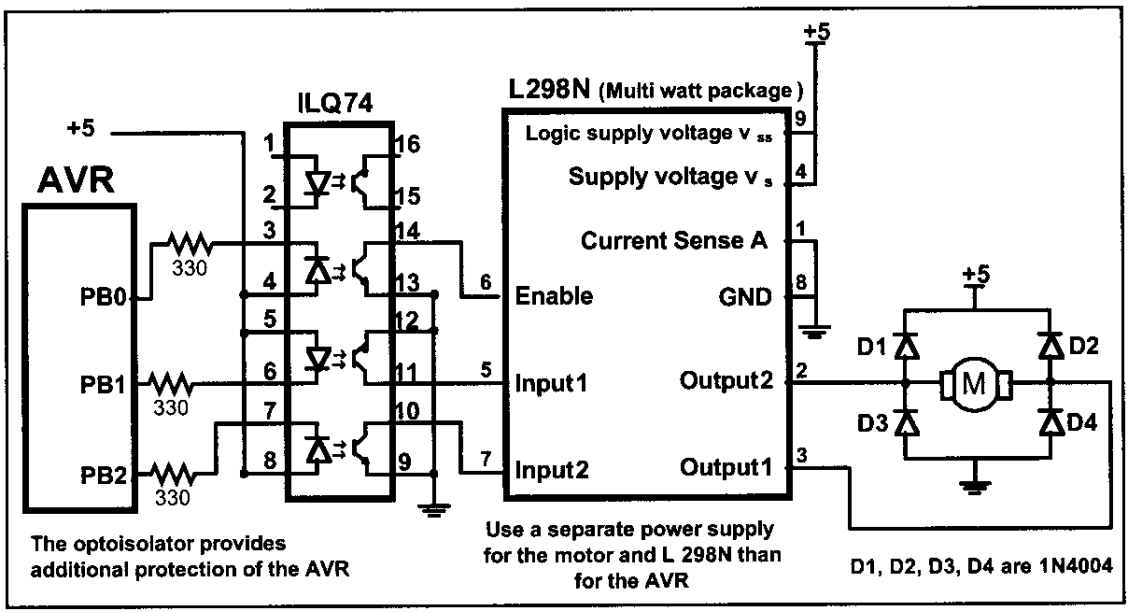


Рис. 4 – Схема реверсивного приводу

Ця програма провадить моніторинг стану перемикача SW і працює за таким алгоритмом:

- якщо SW = 0, то напрям руху «Вперед»;
- якщо SW = 1, то напрям руху «Назад».

Текст програми

```
#include "avr/io.h"

#define ENABLE 0
#define MTR_1 1
#define MTR_2 2
#define SW (PINA&0x80)
int main ( )
{
      DDRA = 0x7F;     //make PA7 input pin
      DDRB = 0xFF;     //make PORTB output pin
      PORTB = PORTB & (~(1<<ENABLE));
      PORTB = PORTB & (~(1<<MTR_1));
      PORTB = PORTB & (~(1<<MTR_2));

      while (1)
      {
            PORTB = PORTB | (1<<ENABLE);

            if(SW == 1)
            {
                  PORTB = PORTB | (1<<MTR_1);       //MTR_1 = 1
                  PORTB = PORTB & (~(1<<MTR_2));    //MTR_2 = 0
            }
            else{
                  PORTB = PORTB & (~(1<<MTR_1)); //MTR_1 = 0
                  PORTB = PORTB | (1<<MTR_2);     //MTR_2 = 1
            }
      }
      return 0;
}
```

Контрольні питання

1. Поясніть поняття «робочий цикл ШІМ»?
2. Вкажіть фактори, які визначають швидкість ДПС.
3. Поясніть режими роботи силового ключа.
4. Поясніть алгоритм програмної реалізації ШІМ.
5. Поясніть основні переваги застосування ШІМ.

Лекція 6

АПАРАТНА РЕАЛІЗАЦІЯ ШИРОТНО-ІМПУЛЬСНОЇ МОДУЛЯЦІЇ

Широко розповсюджений мікроконтролер ATmega32 містить у своєму складі три вбудованих апаратно реалізованих таймери – Timer0, Timer1 та Timer2. Функціональну схему вбудованих таймерів наведено на рис. 1.

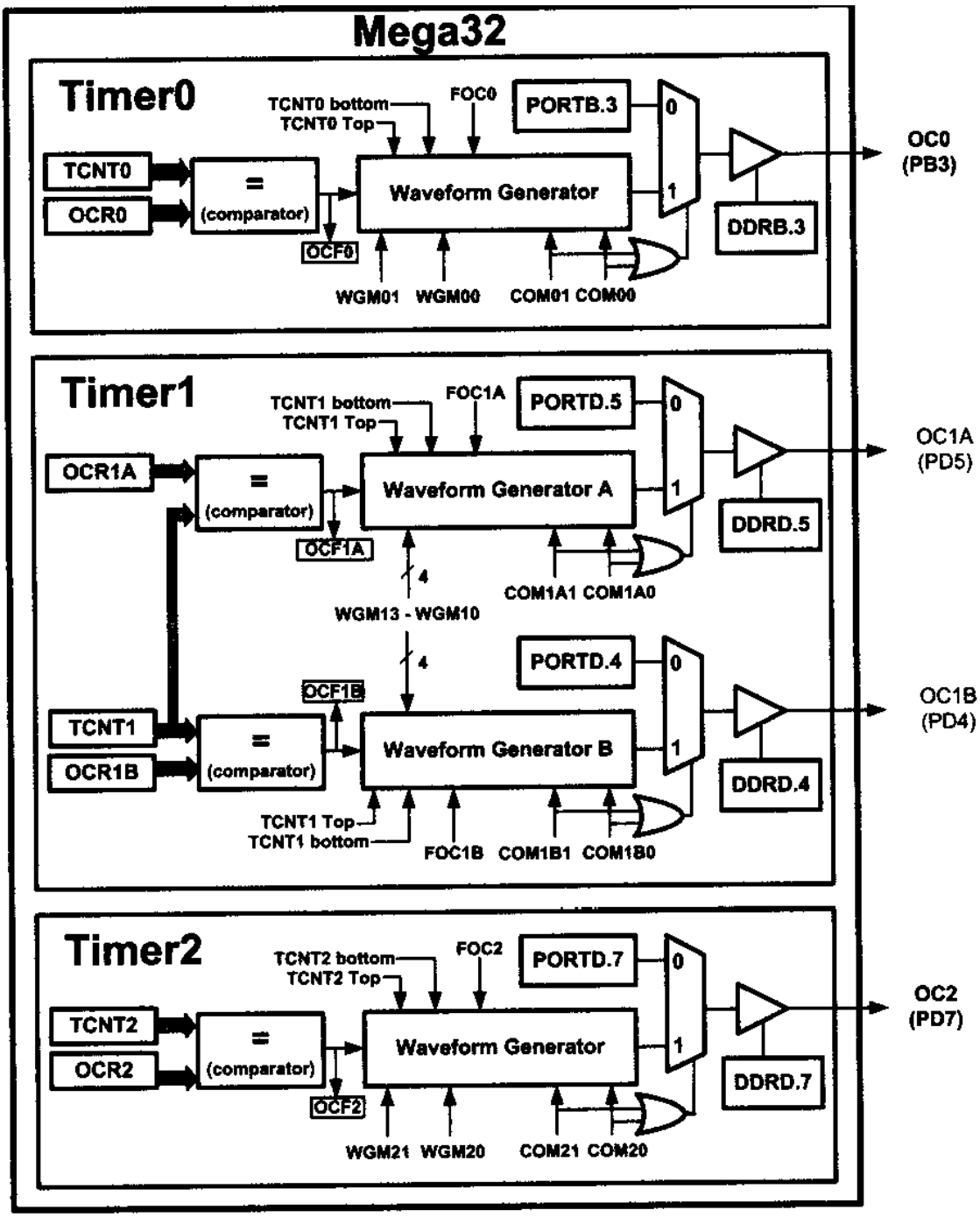


Рис. 1 – Функціональна схема вбудованих таймерів

Раніше ми розглянули програмний метод формування ШІМ-сигналів. Хоч такий метод є цілком працездатним, суттєвим його недоліком є повна занятість процесора власне процесом формування ШІМ-сигналів і неможливість виконувати паралельно інші завдання. Альтернативою такому підходу є формування ШІМ-сигналів за допомогою вбудованих таймерів. В такому разі процесор має лише видати на таймер команду створити ШІМ-сигнал із заданими параметрами, після чого перейти на виконання інших завдань.

Далі розглянемо режими генерації ШІМ-сигналів на базі таймера 0.

Режим «Fast PWM»

В цьому режимі таймер працює подібно до нормального режиму. Після запуску вміст лічильного регістра TCNT0 таймера зростає. Рахунок продовжується до моменту досягнення значення 0xFF. З надходженням ще одного вхідного імпульсу виконується перехід 0xFF -> 0x00, тобто лічильний регістр таймера TCNT0 обнулюється. Цей перехід називається переповненням таймера (рис. 2) . В момент переповнення встановлюється прапор переповнення TOV0 = 1.

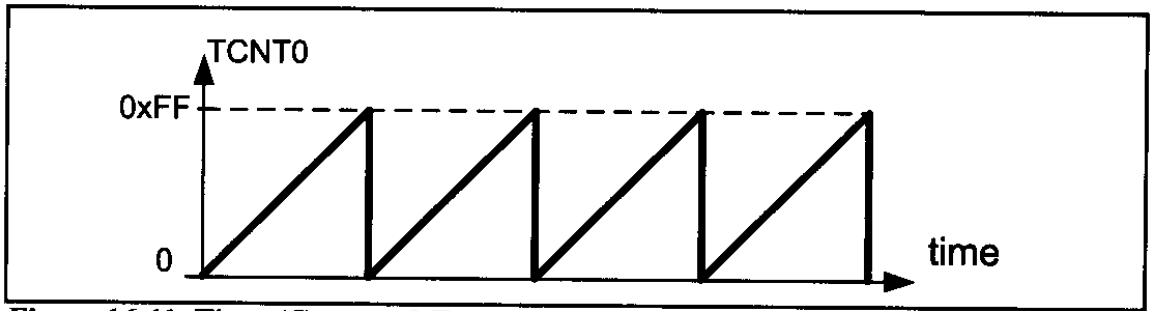


Рис. 2 – Таймер 0 в режимі «Fast PWM»

Керування режимами ШІМ провадиться за допомогою регістра TCCR0, формат якого наведено на рис. 3.

Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

FOC0 D7 Force compare match: it is a write-only bit, which can be used while generating a wave. Writing 1 to it causes the wave generator to act as if a compare match has occurred (see Chapter 15).

WGM01:00 D3D6 Timer0 mode selector bit

0 0 Normal
0 1 PWM, Phase correct
1 0 CTC (Clear Timer on Compare match)
1 1 Fast PWM

COM01:00 D5 D4 Compare Output Mode when Timer0 is in Fast PWM mode

COM01	COM00	Mode Name	Description
0	0	Disconnected	Normal port operation, OC0 disconnected
0	1	Reserved	Reserved
1	0	Non-inverted	Clear OC0 on compare match, set OC0 at TOP
1	1	Inverted PWM	Set OC0 on compare match, clear OC0 at TOP

CS02:00 D2D1D0 Timer0 clock selector

0 0 0 No clock source (Timer/Counter stopped)
0 0 1 clk (no prescaling)
0 1 0 clk / 8
0 1 1 clk / 64
1 0 0 clk / 256
1 0 1 clk / 1024
1 1 0 External clock source on T0 pin. Clock on falling edge
1 1 1 External clock source on T0 pin. Clock on rising edge

Рис. 3 – Регістр керування TCCR0

Якщо COM01:00 = 00, то зовнішній вивід OC0 працює як звичайна лінія порту вводу/виводу.

Якщо COM01:00 = 10, то вбудований генератор коливань скидає сигнал на OC0 в нуль в момент співпадіння і встановлює його в одиницю при досягненні рівня «TOP». Цей режим називається не інвертованою ШІМ (рис. 4). В цьому режимі при збільшенні значення регістру OCR0 тривалість вихідного імпульсу зростає.

Якщо COM01:00 = 11, то вбудований генератор коливань встановлює сигнал на OC0 в одиницю в момент співпадіння і скидає його в нуль при досягненні рівня «TOP». Цей режим називається інвертованою ШІМ (рис. 5).

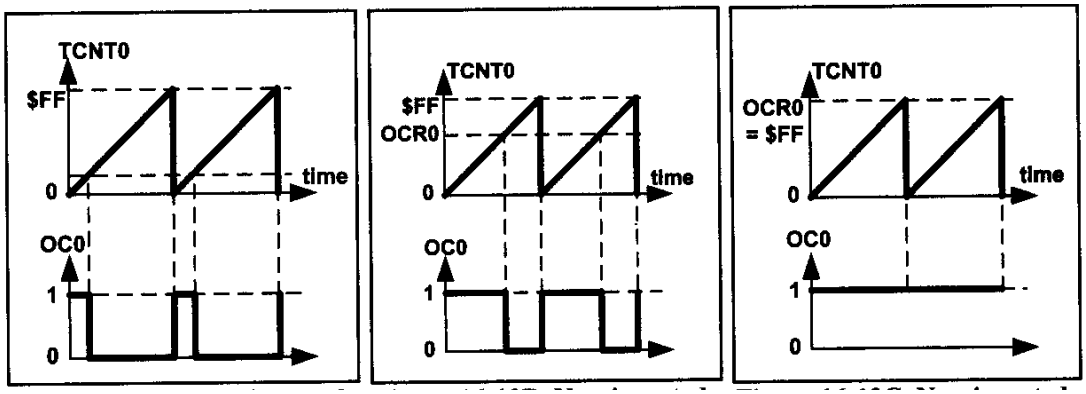


Рис. 4 – Не інвертована ШІМ

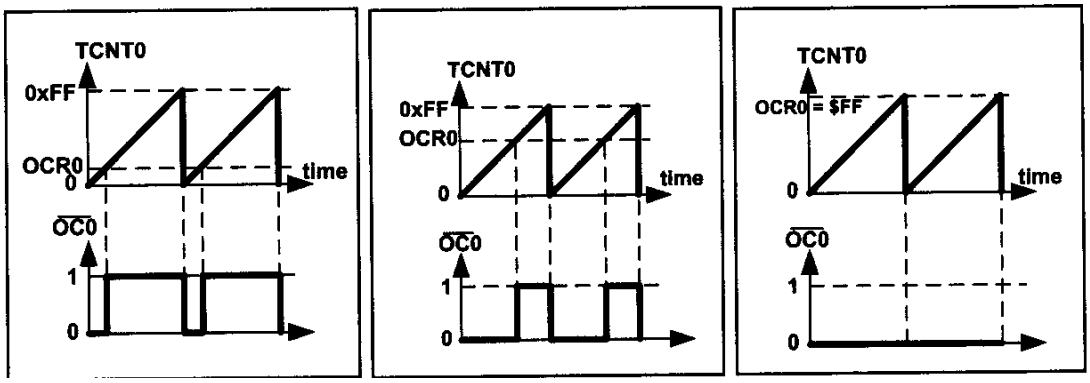


Рис. 3.5 – Інвертована ШІМ

Очевидно, що в режимі інвертованої ШІМ при збільшенні значення регістру OCR0 тривалість вихідного імпульсу зменшується.

Контрольні питання

1. Поясніть, як перевести лінію OC0 в режим звичайної лінії порту.
2. За допомогою якого регістра провадиться керування режимами ШІМ?
3. Поясніть, як перевести лінію OC0 в режим неінвертованої ШІМ.
4. Поясніть режими генерації ШІМ за допомогою таймера 0.
5. Поясніть, як перевести лінію OC0 в режим інвертованої ШІМ.

Лекція 7

РОЗРАХУНОК ПАРАМЕТРІВ ШВИДКОЇ ШІМ

В режимі швидкої ШІМ таймер виконує рахунок вхідних імпульсів від 0 до значення TOP (TOP = 0xFF для восьми бітних таймерів), після досягнення якого процес повторюється. Отже, частота генерованих коливань є 1/256 від частоти вхідних імпульсів таймера. Частоту вхідних імпульсів можна змінювати за допомогою попереднього дільника.

Таким чином, для восьми бітних таймерів частоту генерованих коливань можна розрахувати наступним чином (N – коефіцієнт ділення попереднього дільника):

$$\left.\begin{aligned} F_{\text{generated wave}} &= \frac{F_{\text{timer clock}}}{256} \\ F_{\text{timer clock}} &= \frac{F_{\text{oscillator}}}{N} \end{aligned}\right\} \Rightarrow F_{\text{generated wave}} = \frac{F_{\text{oscillator}}}{256 \times N}$$

Розрахунок робочого циклу генерованих коливань

В режимі швидкої ШІМ робочий цикл генерованих коливань може бути визначений згідно значення регістру OCCR0.

В режимі неінвертованої ШІМ (COM01:00 = 10) більшому значенню в регістрі OCR0 відповідає більший робочий цикл. Нехай значення регістру OCR0 = 255, тоді на виході OC0 буде сформовано вихідний імпульс тривалістю 256 вхідних імпульсів таймера, що становить робочий цикл 100%.

Із цього можна заключити, що тривалість вихідного імпульсу становить «Вміст регістру OCR0 + 1» (рис. 1).

Отже, для режиму неінвертованої ШІМ робочий цикл Duty Cycle (в %) може бути розрахований за формулою:

$$\text{Duty Cycle} = \frac{\text{OCR0} + 1}{256} \times 100$$

Аналогічно, для режиму інвертованої ШІМ робочий цикл Duty Cycle (в %) може бути розрахований за формулою:

$$\text{Duty Cycle} = \frac{255 - \text{OCR0}}{256} \times 100$$

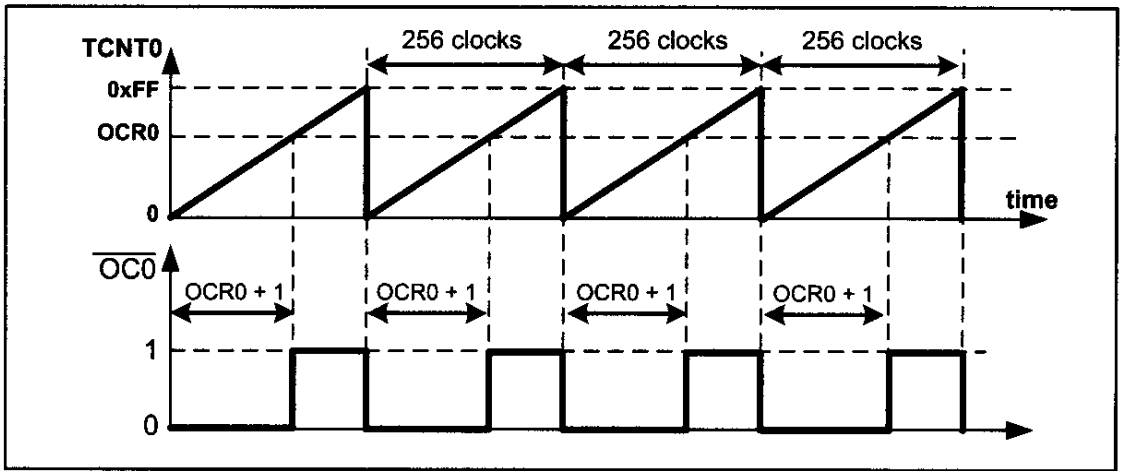


Рис. 1 – Таймер 0 в режимі швидкої ШІМ

Приклад 1
Розрахувати значення OCR0 для генерації в режимі не інвертованої ШІМ робочого циклу 75%
Рішення:
$75 = (\text{OCR0} + 1) \times 100 / 256 \rightarrow \text{OCR0} + 1 = 75 \times 256 / 100 = 192 \rightarrow \text{OCR0} =$

Приклад 2
Знайти значення TCCR0 для налаштування таймеру 0 на режим швидкої неінвертованої ШІМ без попереднього ділення частоти
Рішення:

WGM01:00 = 11 = Fast PWM mode CS02:00 = 001 = No prescaler
COM01:00 = 10 = Non-inverted PWM

TCCR0 =

0	1	1	0	1	0	0	1
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Приклад 3

За умови XTAL = 8 МГц і не інвертованої ШІМ скласти програму, що генерує коливання з частотою 31,250 Гц і робочим циклом 75%

Рішення:

31,250 = 8M / (256 × N) ➔ N = 8M / (31,250 × 256) = 1 ➔ N = 1 ➔ No presca

```
.INCLUDE "M32DEF.INC"
      SBI   DDRB,3
      LDI   R20,191     ;from Example
      OUT   OCR0,R20    ;OCR0 = 191
      LDI   R20,0x69    ;from Example
      OUT   TCCR0,R20   ;Fast PWM, no prescaler, non-inverted
HERE: RJMP  HERE        ;infinite loop
```

Приклад 4

За умови XTAL = 8 МГц і неінвертованої ШІМ скласти програму, що генерує коливання з частотою 3906,25 Гц і робочим циклом 37,5%

Рішення:

3906.25 = 8M / (256 × N) ➔ N = 8M / (3906.25 × 256) = 8 ➔ the prescaler value =
37.5 = 100 × (OCR0 + 1) / 256 ➔ OCR0 + 1 = (256 × 37.5) / 100 = 96 ➔ OCR0 =

```
.INCLUDE "M32DEF.INC"
      SBI   DDRB,3
      LDI   R20,95
      OUT   OCR0,R20    ;OCR0 = 95
      LDI   R20,0x6A
      OUT   TCCR0,R20   ;Fast PWM, N = 8, non-inverted
HERE: RJMP  HERE
```

Приклад 5
Переписати приклад 4.4 для випадку інвертованої ШІМ
Рішення:

$37.5 = 100 \times (255 - OCR0)/256 \rightarrow 255 - OCR0 = (256 \times 37.5)/100 = 96 \rightarrow OCR0 =$

```
.INCLUDE "M32DEF.INC"
      SBI   DDRB,3
      LDI   R20,159
      OUT   OCR0,R20    ;OCR0 = 159
      LDI   R20,0x7A
      OUT   TCCR0,R20   ;Fast PWM, N = 8, inverted
HERE: RJMP  HERE
```

Завантаження регістру OCR0 в режимах ШІМ

В режимах, в яких ШІМ не використовується (CTC та Normal mode), при завантаженні коду в регістр OCR0, він безпосередньо і негайно попадає у вказаний регістр. Але в режимах ШІМ (Fast PWM та Phase correct PWM) так не відбувається, тому що перед власне регістром OCR0 вводиться буферний регістр OCR0 Buffer (рис. 2).

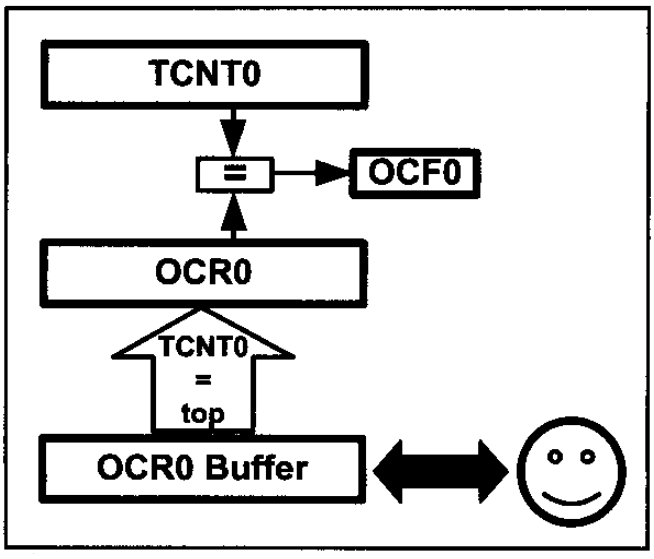


Рис. 2 – Завантаження регістру OCR0

Отже, коли ми завантажуємо (або читаємо) регістр OCR0, насправді ми виконуємо цю операцію з буферним регістром OCR0 Buffer.

Вміст буферного регістру OCR0 Buffer буде завантажений в регістр OCR0, лише в момент, коли значення TCNT0 стане рівним TOP (для восьми бітних таймерів TOP = 0xFF).

Приклад 6

Навести графік коливань, які генеруються наступною програмою.

Прийняти XTALL = 1 МГц.

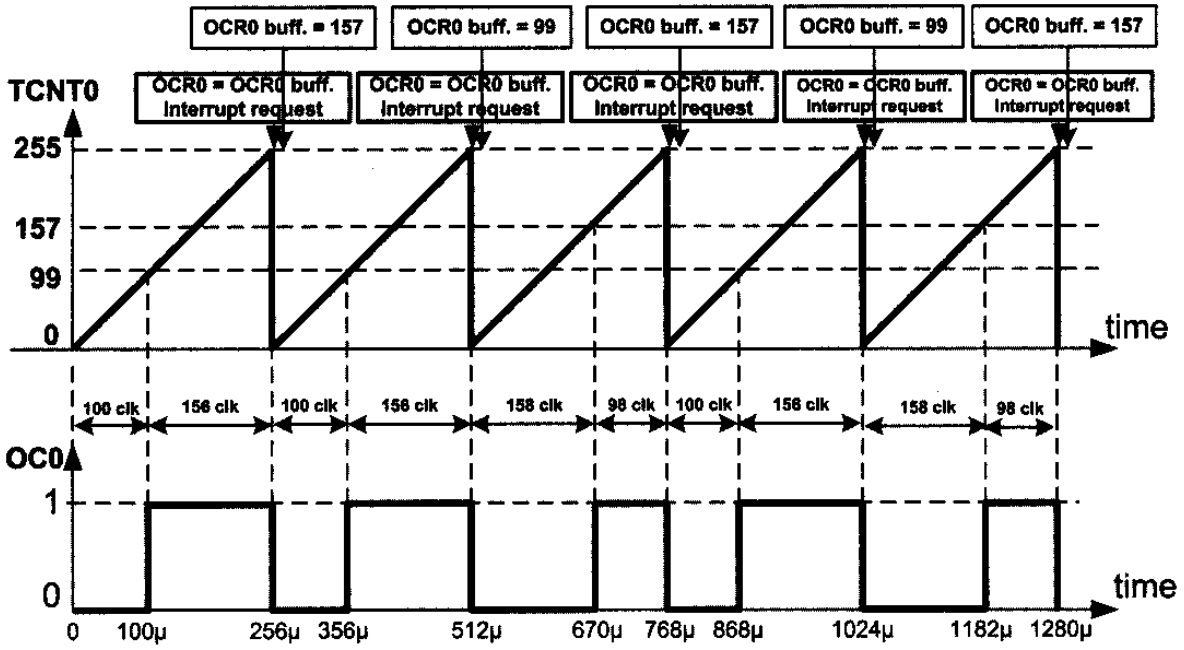
```
      .INCLUDE "M32DEF.INC"
      RJMP  MAIN
      .ORG  0x16              ;Timer0 overflow interrupt vector
      NEG   R20               ;Negative R20
      OUT   OCR0,R20          ;OCR0 = R20
      RETI                    ;return interrupt
MAIN:
      LDI   R16,HIGH(RAMEND)
      OUT   SPH,R16
      LDI   R16,LOW(RAMEND)
      OUT   SPL,R16           ;initialize stack
      SBI   DDRB,3            ;OC0 as output
      LDI   R20,99            ;R20 = 99
      OUT   OCR0,R20          ;OCR0 = 99
      LDI   R16,0x69    ;Fast PWM mode, non-inverted, no prescal
      OUT   TCCR0,R16
      OUT   OCR0,R20          ;OCR0 buffer = 99
      LDI   R16,(1<<TOIE0)    ;enable overflow interrupt
      OUT   TIMSK,R16
      SEI                     ;enable interrupt
HERE: RJMP  HERE              ;wait here
```

Рішення:

Генератор коливань працює в режимі неінвертованої швидкої ШІМ (non-inverted Fast PWM). Це означає, що в момент співпадіння на OC0 буде встановлено високий рівень. Оскільки в регістр OCR0 завантажено значення 99, то співпадіння відбудеться, коли вміст TCNT0 досягне значення 99. В момент переповнення таймера буде згенеровано запит переривання і в буферний регістр OCR0 buffer буде завантажено число 157

(доповнення до двох від 99). За наступного переповнення таймера вміст буферного регістра OCR0 buffer буде завантажено в регістр порівняння OCR0. В момент наступного переповнення таймера відбудеться нове переривання і в регістр OCR0 buffer буде завантажено число 99 (доповнення до двох від 157).

Отже, часові діаграми матимуть вид:



Контрольні питання

1. Який рівень сигналу встановлюється на виводі OC0 в режимі неінвертованої ШІМ?
2. Як розраховується робочий цикл для інвертованої ШІМ?
3. Чому дорівнює значення TOP для восьмибітних таймерів?
4. Яку функцію виконує буферний регістр таймера?
5. Як розраховується робочий цикл для неінвертованої ШІМ?

Лекція 8

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ТА ЗАСТОСУВАННЯ КРОКОВИХ ДВИГУНІВ

Крокові двигуни вже давно й успішно застосовуються в найрізноманітніших пристроях. Їх можна зустріти в дисководах, принтерах, плоттерах, сканерах, факсах, а також у різноманітному промисловому й спеціальному устаткуванні. У цей час випускається безліч різних типів крокових двигунів на всі випадки життя. Однак правильно вибрати тип двигуна - це ще пів-справи. Не менш важливо правильно вибрати схему контролера й алгоритм його роботи, що найчастіше визначається програмою мікроконтролера.

Ціль цього розділу - систематизувати відомості про конструкцію крокових двигунів, способи керування ними, схеми контролерів і алгоритми роботи. Як приклад буде наведена практична реалізація простого й недорогого драйвера крокового двигуна на основі мікроконтролера сімейства AVR.

Кроковий двигун - це електромеханічний пристрій, що перетворює електричні імпульси в дискретні механічні переміщення.

Напевно, багато хто бачив, як виглядає кроковий двигун зовні: він практично нічим не відрізняється від двигунів інших типів. Найчастіше це круглий корпус, вал, декілька виводів (рис. 1).

Рис. 1. Зовнішній вигляд крокових двигунів родини ДШИ-200.

Однак крокові двигуни мають деякі унікальні властивості, що робить часом їх винятково зручними для застосування або навіть незамінними.

Переваги крокових двигунів:

- кут повороту ротора визначається числом імпульсів, які подані на двигун

двигун забезпечує повний момент у режимі зупинки (якщо обмотки мають живлення);

- прецизійне позиціонування й повторюваність. Гарні крокові двигуни мають точність 3-5% від величини кроку. Ця помилка не накопичується від кроку до кроку;
- можливість швидкого старту/зупинки/реверсування;
- висока надійність, пов'язана з відсутністю щіток, термін служби крокового двигуна фактично визначається терміном служби підшипників;
- однозначна залежність положення від вхідних імпульсів забезпечує позиціонування без зворотного зв'язку;
- можливість одержання дуже низьких швидкостей обертання для навантаження, приєднаної безпосередньо до вала двигуна без проміжного редуктора;
- може бути перекритий досить великий діапазон швидкостей, швидкість пропорційна частоті вхідних імпульсів;

Звичайно, кроковим двигунам притаманні і певні недоліки, а саме:

- кроковим двигуном властиве явище резонансу;
- можлива втрата контролю положення внаслідок роботи без зворотного зв'язку;

- споживання енергії не зменшується навіть без навантаження;
- утруднено роботу на високих швидкостях;
- невисока питома потужність;
- відносно складна схема керування.

Підходи до вибору двигунів

Крокові двигуни відносяться до класу безколекторних двигунів постійного струму. Як і будь-які безколекторні двигуни, вони мають високу надійність і великий термін служби, що дозволяє застосовувати їх у критичних, наприклад, індустріальних застосуваннях. У порівнянні зі звичайними двигунами постійного струму, крокові двигуни вимагають значно більш складних схем керування, які повинні виконувати всі комутації обмоток при роботі двигуна. Крім того, сам кроковий двигун - дорогий пристрій, тому там, де точне позиціонування не потрібне, звичайні колекторні двигуни мають помітну перевагу. Справедливості заради слід зазначити, що останнім часом для керування колекторними двигунами всі частіше застосовують контролери, які по складності практично не уступають контролерам крокових двигунів.

Одним з головних переваг крокових двигунів є можливість здійснювати точне позиціонування й регулювання швидкості без датчика зворотного зв'язку. Це дуже важливо, тому що такі датчики можуть коштувати набагато більше самого двигуна. Однак це підходить тільки для систем, які працюють при малому прискоренні й з відносно постійним навантаженням. У той же час системи зі зворотним зв'язком здатні працювати з більшими прискореннями й навіть при змінному характері навантаження. Якщо навантаження крокового двигуна перевищить його момент, то інформація про положення ротора губиться й система вимагає базування за допомогою, наприклад, кінцевого вимикача або іншого датчика. Системи зі зворотним зв'язком не мають подібного недоліку.

При проектуванні конкретних систем доводиться робити вибір між сервоприводом і кроковим приводом. Коли потрібне прецизійне позиціонування й точне керування швидкістю, а необхідний момент і швидкість не виходять за припустимі межі, то кроковий двигун є найбільш економічним рішенням. Як і для звичайних двигунів, для підвищення моменту може бути використаний понижуючий редуктор. Однак для крокових двигунів редуктор не завжди підходить. На відміну від колекторних двигунів, у яких момент росте зі збільшенням швидкості, кроковий двигун має більший момент на низьких швидкостях. До того ж, крокові двигуни мають набагато меншу максимальну швидкість у порівнянні з колекторними двигунами, що обмежує максимальне передаточне число й, відповідно, збільшення моменту за допомогою редуктора. Готові крокові двигуни з редукторами хоча й існують, однак є екзотикою. Ще одним фактом, що обмежує застосування редуктора, є властивий йому люфт.

Можливість одержання низької частоти обертання часто є причиною того, що розробники, будучи не в змозі спроектувати редуктор, застосовують крокові двигуни невиправдано часто. У той же час колекторний двигун має більше високу питому потужність, низьку вартість, просту схему керування, і разом з одноступінчастим черв'ячним редуктором він здатний забезпечити той же діапазон швидкостей, що й кроковий двигун. До того ж, при цьому забезпечується значно більший момент. Приводи на основі колекторних двигунів дуже часто застосовуються в техніці військового призначення, а це побічно говорить про гарні параметри й високу надійність таких приводів. Та й у сучасній побутовій техніці, автомобілях, промисловому устаткуванні колекторні двигуни поширені досить сильно. Проте, для крокових двигунів є своя, хоча й досить вузька, сфера застосування, де вони незамінні.

Контрольні питання

1. Охарактеризуйте переваги крокових двигунів.
2. Особливості вибору сервоприводу та крокового приводу.
3. Поясніть підходи до вибору двигунів.
4. Які фактори визначають високу надійність крокових двигунів?
5. Поясніть недоліки крокових двигунів.

Лекція 9

ВИДИ КРОКОВИХ ДВИГУНІВ

Існують три основних типи крокових двигунів:

- двигуни зі змінним магнітним опором;
- двигуни з постійними магнітами;
- гібридні двигуни.

Визначити тип двигуна можна навіть на дотик: при обертанні вала знеструмленого двигуна з постійними магнітами (або гібридного) почувається змінний опір обертанню, двигун обертається як би щигликами. У той же час вал знеструмленого двигуна зі змінним магнітним опором обертається вільно.

Гібридні двигуни є подальшим удосконаленням двигунів з постійними магнітами й по способу керування нічим від них не відрізняються. Визначити тип двигуна можна також по конфігурації обмоток. Двигуни зі змінним магнітним опором звичайно мають три (рідше чотири) обмотки з одним загальним виводом.

Двигуни з постійними магнітами найчастіше мають дві незалежні обмотки. Ці обмотки можуть мати відводи від середини. Іноді двигуни з постійними магнітами мають 4 роздільні обмотки.

У кроковому двигуні обертаючий момент створюється магнітними потоками статора й ротора, які відповідним чином орієнтовані один відносно одного. Статор виготовлений з матеріалу з високою магнітною проникністю й має кілька полюсів. Полюс можна визначити як деяку область намагніченого тіла, де сконцентроване магнітне поле. Полюси розміщені як на статорі, так і на роторі. Для зменшення втрат на вихрові струми магнітопроводи зібрані з окремих пластин, подібно сердечнику трансформатора. Обертаючий момент пропорційний величині магнітного поля, що пропорційна струму в обмотці й кількості витків. Таким чином,

момент залежить від параметрів обмоток. Якщо хоча б одна обмотка крокового двигуна має живлення, ротор приймає певне положення. Він буде перебувати в цьому положенні доти, поки зовнішній прикладений момент не перевищить деякого значення, яке називається моментом утримання. Після цього ротор повернеться й буде намагатися прийняти одне з наступних положень рівноваги.

Двигуни зі змінним магнітним опором

Крокові двигуни зі змінним магнітним опором мають кілька полюсів на статорі й ротор зубчастої форми з магнітном'якого матеріалу (рис. 2). Намагніченість ротора відсутня. Для простоти на рисунку ротор має 4 зубці, а статор має 6 полюсів. Двигун має 3 незалежні обмотки, кожна з яких намотана на двох протилежних полюсах статора. Величина кроку такого двигуна становить 30 градусів.

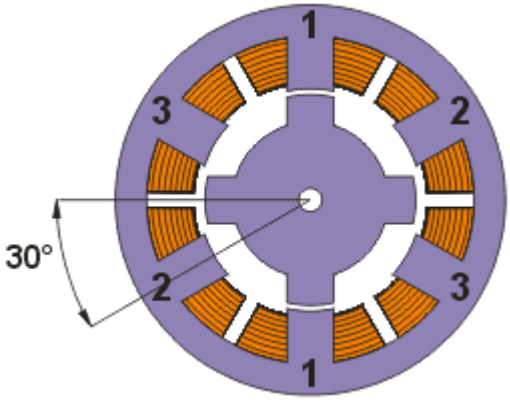


Рис. 2. Двигун зі змінним магнітним опором.

При включенні струму в одній з котушок, ротор прагне зайняти положення, коли магнітний потік замкнутий, тобто зубець ротора буде перебувати напроти тих полюсів, на яких перебуває обмотка зі струмом.

Якщо потім виключити цю обмотку й включити наступну, то ротор поміняє положення, знову замкнувши своїми зубцями магнітний потік.

Таким чином, щоб здійснити безперервне обертання, потрібно вмикати фази поперемінно. Двигун не чутливий до напрямку струму в обмотках. Реальний двигун може мати більшу кількість полюсів статора й більшу кількість зубців ротора, що відповідає більшій кількості кроків на оберт. Іноді поверхню кожного полюса статора виконують зубчастою, що разом з відповідними зубцями ротора забезпечує дуже маленьке значення кута кроку, порядку декількох градусів.

Двигуни зі змінним магнітним опором досить рідко використовують в індустріальних застосуваннях.

Двигуни з постійними магнітами

Двигуни з постійними магнітами складаються зі статора, що має обмотки, і ротора, що містить постійні магніти (рис.. 3). Полюси ротора мають прямолінійну форму й розташовані паралельно осі двигуна.

Завдяки намагніченості ротора в таких двигунах забезпечується більший магнітний потік і, як наслідок, більший момент, порівняно з двигунами зі змінним магнітним опором.

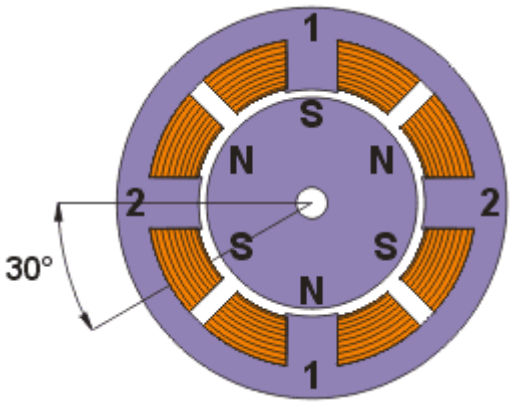


Рис. 3. Двигун з постійними магнітами.

Показаний на рисунку двигун має 3 пари полюсів ротора й 2 пари полюсів статора. Двигун має дві незалежні обмотки, кожна з яких намотана на двох протилежних полюсах статора. Такий двигун, як і розглянутий раніше двигун зі змінним магнітним опором, має величину кроку 30 град. При наявності струму в одній з котушок, ротор прагне зайняти таке положення, коли різнойменні полюси ротора й статора перебувають один напроти одного. Для здійснення безперервного обертання потрібно включати фази попеременно. На практиці двигуни з постійними магнітами звичайно мають 48 - 24 кроків на оберт (кут кроку 7.5 - 15 град).

Розріз реального крокового двигуна з постійними магнітами показаний на
рис. 4.

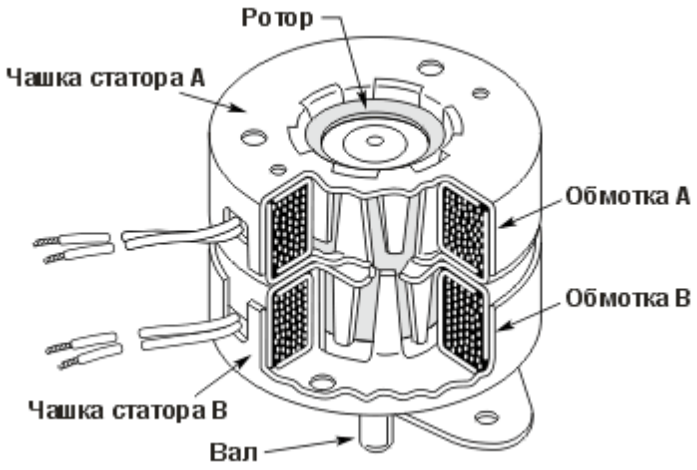


Рис. 4. Розріз крокового двигуна з постійними магнітами

Для здешевлення конструкції двигуна магнітопровод статора виконаний у вигляді штампованої склянки. Усередині перебувають полюсні наконечники у вигляді ламелів. Обмотки фаз розміщені на двох різних

магнітопроводах, які встановлені у корпусі впритул один до одного. Ротор являє собою циліндричний багатополюсний постійний магніт.

Двигуни з постійними магнітами піддаються впливу зворотної ЭДС із боку ротора, яка обмежує максимальну швидкість. Для роботи на високих швидкостях застосовують двигуни зі змінним магнітним опором.

Контрольні питання

1. Наведіть основні типи крокових двигунів.
2. Поясніть конструкцію ротора крокового двигуна з постійними магнітами.
3. Яку кількість кроків на оберт мають крокові двигуни з постійними магнітами?
4. Поясніть конструкцію крокового двигуна з постійними магнітами.
5. Яку кількість кроків на оберт мають крокові двигуни зі змінним магнітним опором?
6. Поясніть конструкцію крокового двигуна зі змінним магнітним опором.

Лекція 10

ГІБРИДНІ ДВИГУНИ

Гібридні двигуни є більше дорогими, порівняно з двигунами з постійними магнітами, зате вони забезпечують меншу величину кроку, більший момент і більшу швидкість. Типове число кроків на оберт для гібридних двигунів становить від 100 до 400 (кут кроку 3.6 - 0.9 град.). Гібридні двигуни сполучають у собі кращі риси двигунів зі змінним магнітним опором і двигунів з постійними магнітами. Ротор гібридного двигуна має зубці, розташовані в осьовому напрямку (рис. 1).

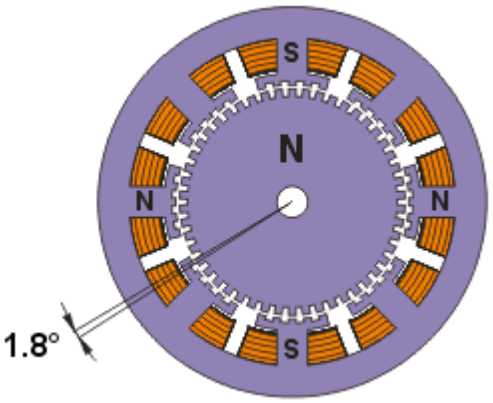


Рис. 1. Гібридний двигун.

Ротор двигуна розділений на дві частини, між якими розташований циліндричний постійним магніт. Таким чином, зубці верхньої половини ротора є північними полюсами, а зубці нижньої половини - південними. Крім того, верхня й нижня половини ротора повернені один відносно другого на половину кута кроку зубців. Число пар полюсів ротора дорівнює кількості зубців на одній з його половинок. Зубчасті полюсні наконечники ротора, як і статор, набрані з окремих пластин для зменшення втрат на вихрові струми.

Статор гібридного двигуна також має зубці, забезпечуючи велику кількість еквівалентних полюсів, на відміну від основних полюсів, на яких

розташовані обмотки. Звичайно використовуються 4 основні полюси для 3.6 градусних двигунів і 8 основних полюсів для 1.8 і 0.9 градусних двигунів. Зубці ротора забезпечують менший опір магнітного ланцюга в певних положеннях ротора, що поліпшує статичний і динамічний момент. Це забезпечується відповідним розташуванням зубців, коли частина зубців ротора перебуває строго напроти зубців статора, а частина між ними.

Залежність між числом полюсів ротора, числом еквівалентних полюсів статора й числом фаз визначає кут кроку S двигуна:

$S = 360/(Nph*Ph) = 360/N,$

де Nph - число еквівалентних полюсів на фазу = число полюсів ротора;

Ph - число фаз;

N - повна кількість полюсів для всіх фаз разом.

Ротор показаного на малюнку двигуна має 100 полюсів (50 пар), двигун має 2 фази, тому повна кількість полюсів - 200, а крок, відповідно, 1.8 град.

Поздовжній розріз гібридного крокового двигуна показано на рис. 2. Стрілками показаний напрямок магнітного потоку постійного магніту ротора. Частина потоку (на малюнку показана чорною лінією) проходить через полюсні наконечники ротора, повітряні зазори й полюсний наконечник статора. Ця частина магнітопроводу не бере участь у створенні моменту.

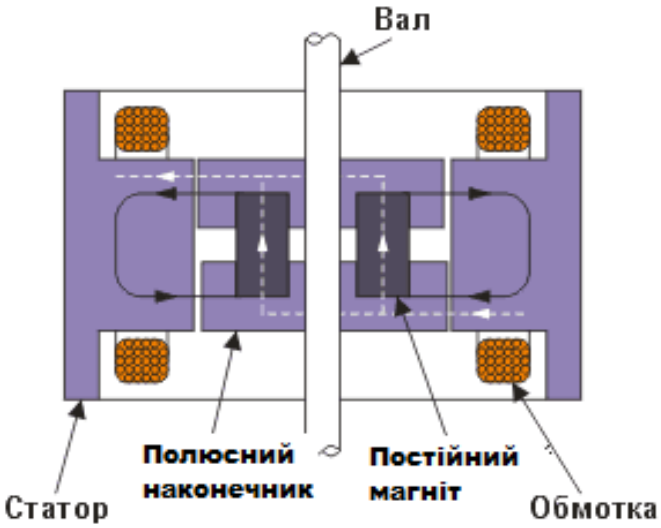


Рис. 2. Поздовжній розріз гібридного крокового двигуна.

Як видно на рисунку, повітряні зазори у верхнього й нижнього полюсного наконечника ротора різні. Це досягається завдяки повороту полюсних наконечників на половину кроку зубів. Тому існує інший магнітний ланцюг, що містить мінімальні повітряні зазори й, як наслідок, має мінімальний магнітний опір. По цьому ланцюзі замикається інша частина потоку (на малюнку показана штриховою білою лінією), яка і створює момент. Частина ланцюга лежить у площині, перпендикулярній рисунку, тому не показана. У цій же площині створюють магнітний потік котушки статора. У гібридному двигуні цей потік частково замикається полюсними наконечниками ротора і постійний магніт його «відчуває» слабко. Тому на відміну від двигунів постійного струму, магніт гібридного двигуна неможливо розмагнітити ні при якій величині струму обмоток.

Величина зазору між зубцями ротора й статора дуже невелика - типово 0.1 мм. Це вимагає високої точності при збиранні двигуна, тому кроковий двигун не варто розбирати заради задоволення цікавості, інакше на цьому його термін служби може закінчитися.

Щоб магнітний потік не замикався через вал, що проходить всередині магніту, його виготовляють із немагнітних марок стали. Вони звичайно

мають підвищену крихкість, тому з валом, особливо малого діаметра, варто поводитися з обережністю.

Для одержання більших моментів необхідно збільшувати як поле, створюване статором, так і поле постійного магніту. При цьому потрібен більший діаметр ротора, що погіршує відношення крутного моменту до моменту інерції. Тому потужні крокові двигуни іноді конструктивно виконують із декількох секцій у вигляді етажерки. Крутний момент і момент інерції збільшуються пропорційно кількості секцій, а їхнє відношення не погіршується.

Існують і інші конструкції крокових двигунів. Наприклад, двигуни з дисковим намагніченим ротором. Такі двигуни мають малий момент інерції ротора, що в ряді випадків є важливим.

Більшість сучасних крокових двигунів є гібридними. По суті гібридний двигун є двигуном з постійними магнітами, але з більшим числом полюсів. По способу керування такі двигуни однакові, далі будуть розглядатися тільки такі двигуни. Найчастіше на практиці двигуни мають 100 або 200 кроків на оберт, відповідно крок дорівнює 3.6 градусів або 1.8 градусів. Більшість контролерів дозволяють працювати в напівкроковому режимі, де цей кут удвічі менше, а деякі контролери забезпечують мікрокроковий режим.

Біполярні й уніполярні крокові двигуни

Залежно від конфігурації обмоток двигуни діляться на біполярні й уніполярні. Біполярний двигун має одну обмотку в кожній фазі, до якої контролер підводить напругу змінної полярності. Для такого типу двигуна потрібен мостовий драйвер, або напівмостовий із двополярним живленням. Всього біполярний двигун має дві обмотки й, відповідно, чотири виводи (рис. 3а).

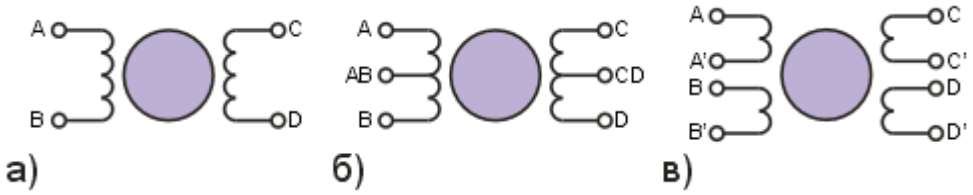


Рис. 3. Біполярний двигун (а), уніполярний (б) і чотириобмоточний (в).

Уніполярний двигун також має одну обмотку в кожній фазі, але від середини обмотки зроблений відвід. Це дозволяє змінювати напрямок магнітного поля, створюваного обмоткою, простим перемиканням половинок обмотки. При цьому істотно спрощується схема драйвера. Драйвер повинен мати тільки 4 простих ключі.

Таким чином, в уніполярному двигуні використовується інший спосіб зміни напрямку магнітного поля. Середні виводи обмоток можуть бути об'єднані усередині двигуна, тому такий двигун може мати 5 або 6 виводів (рис. 3б).

Іноді уніполярні двигуни мають роздільні 4 обмотки, із цієї причини їх помилково називають 4-х фазними двигунами. Кожна обмотка має окремі виводи, тому всього виводів 8 (мал. 3в). При відповідному з'єднанні обмоток такий двигун можна використати як уніполярний або як біполярний. Уніполярний двигун із двома обмотками й відводами теж можна використати в біполярному режимі, якщо відводи залишити непідключеними. У кожному разі струм обмоток варто вибирати так, щоб не перевищити максимально припустиму температуру двигуна .

Порівняння біполярного та уніполярного двигунів

Якщо порівнювати між собою біполярний і уніполярний двигуни, то біполярний має більше високу питому потужність. При тих самих розмірах біполярні двигуни забезпечують більший момент.

Момент, створюваний кроковим двигуном, пропорційний величині магнітного поля, створюваного обмотками статора. Шлях для підвищення магнітного поля - це збільшення струму або числа витків обмоток. Природним обмеженням при підвищенні струму обмоток є фактор насичення залізного сердечника. Однак на практиці це обмеження діє рідко.

Набагато більше істотним є обмеження по нагріванню двигуна внаслідок омічних втрат в обмотках. Саме цей факт і демонструє одну з переваг біполярних двигунів. В уніполярному двигуні в кожний момент часу використовується лише половина обмоток. Інша половина просто займає місце у вікні сердечника, що змушує робити обмотки проводом меншого діаметра.

У той же час у біполярному двигуні завжди працюють всі обмотки, тобто їхнє використання оптимальне. У такому двигуні перетин окремих обмоток удвічі більше, а омічний опір - відповідно вдвічі менше. Це дозволяє збільшити струм у корінь із двох разів при тих же втратах, що дає виграш у моменті приблизно 40%. Якщо ж підвищеного моменту не потрібно, уніполярний двигун дозволяє зменшити габарити або просто працювати з меншими втратами.

На практиці все-таки часто застосовують уніполярні двигуни, тому що вони вимагають значно більше простих схем керування. Це важливо, якщо драйвери виконані на дискретних компонентах.

На даний час існують спеціалізовані мікросхеми драйверів для біполярних двигунів, з використанням яких драйвер виходить не складніше, ніж для уніполярного двигуна. Наприклад, це мікросхеми L293E, L298N або L6202 фірми SGS-Thomson, PBL3770, PBL3774 фірми Ericsson, NJM3717, NJM3770, NJM3774 фірми JRC, A3957 фірми Allegro, LMD18T245 фірми National Semiconductor.

Контрольні питання

1. Поясніть переваги біполярних двигунів перед уніполярними.
2. Які фактори визначають кут кроку двигуна?
3. Наведіть варіанти виконання обмоток крокових двигунів.
4. Яка величина зазору між зубцями ротора та статора?
5. Поясніть фактори, що обмежують момент крокового двигуна.

Лекція 11

ОСОБЛИВОСТІ РЕЖИМІВ КРОКОВИХ ДВИГУНІВ

У порівнянні з повнокроковим режимом напівкроковий режим має наступні переваги:

- більш висока роздільна здатність без застосування двигунів більш високої вартості;
- менші проблеми з явищем резонансу. Резонанс призводить лише до часткової втрати моменту, що звичайно не заважає нормальній роботі приводу.

Недоліком напівкрокового режиму є досить значне коливання моменту від кроку до кроку. У тих положеннях ротора, коли підключена одна фаза, момент становить приблизно 70% від повного значення моменту, коли підключені дві фази. Ці коливання можуть стати причиною підвищених вібрацій і шуму, хоча вони однаково залишаються меншими порівняно з повнокроковим режимом.

Способом усунення коливань моменту є підняття моменту в положеннях з однією включеною фазою й забезпечення в такий спосіб однакового моменту у всіх положеннях ротора. Це може бути досягнуте шляхом збільшення струму в цих положеннях до рівня приблизно 141% від номінального. Деякі драйвери, такі як PBL 3717/2 і PBL 3770A фірми Ericsson, мають логічні входи для зміни величини струму.

Потрібно зазначити, що величина 141% є теоретичною, тому в застосуваннях, що вимагають високої точності підтримки моменту ця величина повинна бути підібрана експериментально для конкретної швидкості й конкретного двигуна. Оскільки струм збільшується тільки в моменти, коли включена одна фаза, розсіювана потужність дорівнює потужності в повнокроковому режимі при струмі 100% від номінального.

Однак таке збільшення струму вимагає більш високої напруги живлення, що не завжди можливо.

Є й інший підхід. Для усунення коливань моменту при роботі двигуна в напівкроковому режимі можна знижувати струм у ті моменти, коли включені дві фази. Для одержання незмінного моменту цей струм повинен становити 70,7% від номінального. У такий спосіб реалізує напівкроковий режим, наприклад, мікросхема драйвера A3955 фірми Allegro.

Для напівкрокового режиму дуже важливим є перехід у стан з однією виключеною фазою. Щоб змусити ротор зайняти відповідне положення, струм у відключеній фазі повинен бути зменшений до нуля якнайшвидше. Тривалість спаду струму залежить від напруги на обмотці в той час, коли вона розсіює свою запасену енергію. Замикаючи в цей час обмотку на джерело живлення, можна забезпечити максимально швидкий спад струму.

Для одержання швидкого спаду струму при живленні обмоток двигуна Н-мостом всі транзистори повинні закриватися, при цьому обмотка через діоди виявляється підключеної до джерела живлення. Швидкість спаду струму значно зменшиться, якщо один транзистор мосту залишити відкритим і замкнути обмотку на транзистор і діод.

Для збільшення швидкості спаду струму при керуванні уніполярними двигунами обмеження викидів ЕРС самоіндукції більш доцільно здійснювати не діодами, а варисторами або комбінацією діодів і стабілітрона, які обмежують викид на більшому, але безпечному для транзисторів рівні.

Мікрокроковий режим

Мікрокроковий режим забезпечується шляхом формування поля статора, що обертається, більш плавно, ніж у повно- або напівкроковому режимах. В результаті забезпечуються менші вібрації й практично

безшумна робота аж до нульової частоти. До того ж менший кут кроку здатний забезпечити більш точне позиціонування.

Існує багато різних мікрокрокових режимів, з величиною кроку від 1/3 до 1/32 повного кроку і навіть менше. Крововий двигун є синхронним електродвигуном. Це означає, що положення рівноваги нерухомого ротора збігається з напрямком магнітного поля статора. При повороті поля статора ротор теж повертається, прагнучи зайняти нове положення рівноваги.

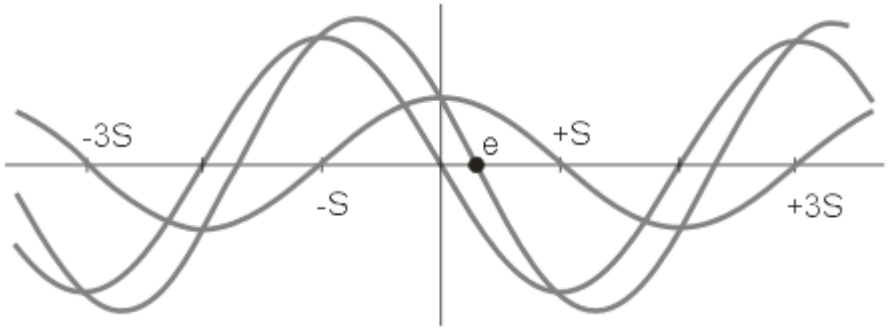


Рис. 1. Залежність моменту від кута повороту ротора
для випадку різних значень струму фаз

Щоб одержати потрібний напрямок магнітного поля, необхідно вибрати не тільки правильний напрямок струмів у обмотках, але й правильне співвідношення цих струмів.

Якщо одночасно підключені дві обмотки двигуна, але струми в цих обмотках не рівні (мал. 1), то результуючий момент буде

Th = (a2 + b2)0.5,

а точка рівноваги ротора зміститься в точку

x = (S / (pi/2)) arctan(b / a),

де a і b - момент, створюваний першою й другою фазою відповідно,

Th - результуючий момент утримання,

x - положення рівноваги ротора в радіанах,

S - кут кроку в радіанах.

Зсув точки рівноваги ротора говорить про те, що ротор можна зафіксувати в будь-якій довільній позиції. Для цього потрібно лише правильно встановити співвідношення струмів у фазах. Саме цей факт використовується при реалізації мікрокрокового режиму.

Ще раз потрібно відзначити, що наведені вище формули вірні тільки в тому випадку, якщо залежність моменту від кута повороту ротора синусоїдальна і якщо жодна частина магнітного кола двигуна не насичується.

Звичайно, кроковий двигун може працювати як синхронний електродвигун у режимі безперервного обертання. Для цього струми його фаз повинні бути синусоїдальними, зміщеними відносно один одного на 90 градусів.

Результатом використання мікрокрокового режиму є набагато більш плавне обертання ротора на низьких частотах. На частотах в 2 - 3 рази вище власної резонансної частоти ротора й навантаження, мікрокроковий режим дає незначні переваги в порівнянні з напівкроковим або повнокроковим режимами. Причиною цього є фільтруюча дія інерції ротора й навантаження. Система із кроковим двигуном працює подібно фільтру нижніх частот.

У мікрокроковому режимі можна здійснювати тільки розгін і гальмування, а основний час працювати в повнокроковомум режимі. До того ж, для досягнення високих швидкостей у мікрокроковому режимі потрібно дуже висока частота повторення мікрокроків, що не завжди може забезпечити керуючий мікроконтролер.

Для запобігання перехідних процесів і втрати кроків, перемикання режимів роботи двигуна (з мікрокрокового режиму в полношаговый і т.п.) необхідно робити в ті моменти, коли ротор перебуває в положенні, що

відповідає одній включеній фазі. Деякі мікросхеми драйверів мікрокрокового режиму мають спеціальний сигнал, що інформує про таке положення ротора. Наприклад, це драйвер A3955 фірми Allegro.

У багатьох практичних застосуваннях, де потрібні малі відносні переміщення й висока роздільна здатність, мікрокроковий режим може замінити механічний редуктор. Часто простота системи є вирішальним чинником, навіть якщо при цьому доведеться застосувати двигун більших габаритів.

Незважаючи на те, що драйвер, який забезпечує мікрокроковий режим, набагато складніший звичайного драйвера, однак система може виявитися більш простою і дешевою, порівняно з кроковим двигуном плюс редуктор.

Сучасні мікроконтролери мають вбудовані цифро-аналогові перетворювачі, які можна використовувати для реалізації мікрокрокового режиму замість спеціальних контролерів. Це дозволяє зробити практично однаковою вартість устаткування для повнокрокового та мікрокрокового режимів.

Іноді мікрокроковий режим використовується для збільшення точності величини кроку понад заявленої виробником двигуна. При цьому використовується номінальне число кроків. Для підвищення точності застосовується корекція положення ротора в точках рівноваги. Для цього спочатку знімають характеристику для конкретного двигуна, а потім, змінюючи співвідношення струмів у фазах, коректують положення ротора індивідуально для кожного кроку.

Такий метод вимагає попереднього калібрування й додаткових ресурсів керуючого мікроконтролера. Крім того, потрібен датчик початкового положення ротора для синхронізації його положення з таблицею коригувальних коефіцієнтів.

На практиці при здійсненні кожного кроку ротор не відразу зупиняється в новому положенні рівноваги, а здійснює загасаючі коливання навколо положення рівноваги. Час встановлення залежить від характеристик навантаження й від схеми драйвера. У більшості застосувань такі коливання є небажаними. Позбутися від цього явища можна шляхом використання мікрокрокового режиму.

На рис. 2 показані переміщення ротора при роботі в повнокроковому та мікрокроковому режимах.

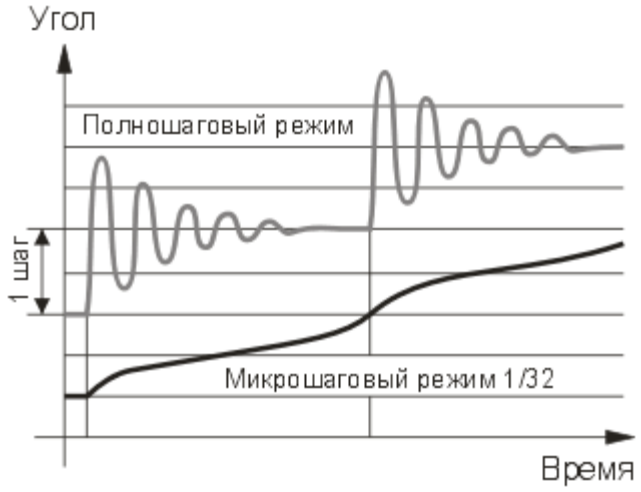


Рис. 2. Переміщення ротора в повнокроковому та мікрокроковому режимах

Видно, що в повнокроковому режимі спостерігаються викиди й коливання, у той час як у мікрокроковому режимі їх немає. Однак і в цьому режимі графік положення ротора відрізняється від прямої лінії. Ці похибки пояснюється похибками геометрії деталей двигуна й можуть бути зменшені за рахунок проведення калібрування й наступної компенсації з метою коректування струмів фаз.

На практиці існують деякі фактори, що обмежують точність роботи приводу в мікрокроковому режимі. Деякі з них пов’язані з драйвером, а інші безпосередньо з двигуном.

Звичайно виробники крокових двигунів вказують такий параметр, як точність кроку. Точність кроку вказується для положень рівноваги ротора при двох включених фазах, струми яких рівні. Це відповідає повнокроковому режиму з перекриттям фаз. Для мікрокрокового режиму, коли струми фаз не рівні, ніяких даних звичайно не приводиться.

Контрольні питання

1. Яку величину кроку має двигун у мікрокроковому режимі?
2. Поясніть способи зменшення коливань моменту.
3. Як забезпечити роботу крокового двигуна у синхронному режимі?
4. Як зафіксувати ротор двигуна у довільній позиції?
5. Які фактори обмежують точність у мікрокроковому режимі?

Лекція 12

МОМЕНТ УТРИМАННЯ І РОБОЧИЙ МОМЕНТ КРОКОВОГО ДВИГУНА

Ідеальний кроковий двигун при живленні фаз синусоїдальним і косинусоїдальным струмом повинен обертатися з постійною швидкістю. У реального двигуна в такому режимі будуть спостерігатися деякі коливання швидкості. Зв'язано це з нестабільністю повітряного зазору між полюсами ротора й статора, наявністю магнітного гістерезису, що призводить до похибок величини й напрямку магнітного поля й т.п. Тому положення рівноваги й момент мають деякі відхилення. Ці відхилення залежать від похибок форми зубців ротора й статора та від застосованого матеріалу магнітопроводів.

Конструкція більшості двигунів оптимізована для найкращої точності в повнокроковому режимі й забезпечення максимального моменту утримання. Спеціальна форма зубців ротора й статора спроектована так, щоб у положенні рівноваги для повно крокового режиму магнітний потік значно зростав. Це приводить до погіршення точності в мікрокроковому режимі. Кращі результати дозволяють одержати двигуни, у яких момент утримання в знеструмленому стані менше.

Відхилення можна розділити на два види: відхилення величини магнітного поля, які приводять до відхилень моменту утримання в мікрокроковому режимі й відхилення напрямку магнітного поля, які приводять до відхилень положення рівноваги. Відхилення моменту утримання в мікрокроковому режимі звичайно становлять 10 - 30% від максимального моменту. Потрібно сказати, що й у повнокроковому режимі момент утримання може коливатися на 10 - 20 % внаслідок недосконалості геометрії ротора й статора.

Якщо виміряти положення рівноваги ротора при обертанні двигуна по й проти годинникової стрілки, то вийдуть трохи різні результати. Цей гістерезис зв'язаний у першу чергу з магнітним гістерезисом матеріалу сердечника, хоча свій внесок вносить і тертя. Магнітний гістерезис приводить до того, що магнітний потік залежить не тільки від струму обмоток, але й від попереднього його значення. Похибка, створювана гістерезисом може досягати величини, що дорівнює декільком мікрокрокам. Тому у високоточних застосуваннях при русі в одному з напрямків потрібно проходити за бажану позицію, а потім повертатися назад, щоб підхід до потрібної позиції завжди здійснювався в одному напрямку.

Цілком природно, що будь-яке бажане збільшення розв'язної здатності натрапляє на якісь фізичні обмеження. Не варто думати, що точність позиціонування для двигуна з кроком 7.2 град. в мікрокроковому режимі не поступається точності двигуна з кроком 1.8 град..

Перешкодою є наступні фізичні обмеження:

- наростання моменту залежно від кута повороту в 7.2 градусного двигуна

  в чотири рази більш пологе, чим у сьогодення 1.8-градусного двигуна. Внаслідок дії моменту тертя або моменту інерції навантаження точність позиціювання вже буде гірше;
- як буде показано нижче, якщо в системі є тертя, то внаслідок появи мертвих зон точність позиціонування буде обмежена;
- більшість комерційних двигунів не мають прецизійну конструкцію й залежність між моментом і кутом повороту ротора не є в точності синусоїдальної. Внаслідок цього залежність між фазою синусоїдального струму живлення й кутом повороту вала буде нелінійною. В результаті ротор двигуна буде точно проходити

положення кожного кроку й півкроку, а між цими положеннями будуть спостерігатися досить значні відхилення.

Ці проблеми найбільше яскраво виражені для двигунів з більшою кількістю полюсів. Існують однак двигуни, ще на етапі розробки оптимізовані для роботи в мікрокроковому режимі. Полюси ротора й статора таких двигунів менш виражені завдяки скошеній формі зубців.

Ще одне джерело похибок позиціонування - це помилка квантування ЦАП, за допомогою якого формуються струми фаз. Справа в тому, що струм повинен формуватися за синусоїдальним законом, тому для мінімізації похибки лінійний ЦАП повинен мати підвищену розрядність. Існують спеціалізовані драйвери з вбудованим нелінійним ЦАП, що дозволяє відразу одержувати відліки функції синуса.

Прикладом може служити драйвер A3955 фірми Allegro, що має вбудований 3-х розрядний ЦАП, який забезпечує наступні значення струму фаз: 100%, 92.4%, 83.1%, 70.7%, 55.5%, 38.2%, 19.5%, 0%. Це дозволяє працювати в мікрокроковому режимі з величиною кроку 1/8, при цьому похибка установки струму фаз не перевищує 2%. Крім того, цей драйвер має можливість управляти швидкістю спаду струму обмоток двигуна під час роботи, що дозволяє зробити «тонке підстроювання» драйвера під конкретний двигун для одержання найменшої погрішності позиціонування.

Навіть якщо ЦАП точно сформував синусоїдальну опорну напругу, цю напругу необхідно підсилити й перетворити в синусоїдальний струм обмоток. Багато драйверів мають значну нелінійність поблизу нульового значення струму, що викликає значні спотворення форми струму і, як наслідок, значні помилки позиціонування. В разі використання високоякісних драйверів, наприклад PBM3960 і PBL3771 фірми Ericsson, похибка, пов'язана із драйвером є дуже малою в порівнянні з похибкою власне двигуна.

Іноді контролери крокових двигунів дозволяють коректувати форму вихідного сигналу шляхом додавання або вирахування із синуса його третьої гармоніки. Однак таке підстроювання повинна виробляється індивідуально під конкретний двигун, характеристики якого повинні бути перед цим відомі.

Через ці обмеження мікрокроковий режим використовується в основному для забезпечення плавного обертання (особливо на дуже низьких швидкостях), для усунення шуму і явища резонансу. Мікрокроковий режим також здатний зменшити час установлення механічної системи, тому що на відміну від повнокрокового режиму відсутні викиди й коливання. Однак у більшості випадків для звичайних двигунів не можна гарантувати точного позиціонування в мікрокроковому режимі.

Синусоїдальний струм фаз може бути забезпечений застосуванням спеціальних драйверів. Деякі з них, наприклад A3955, A3957 фірми Allegro, уже містять ЦАП і вимагають від мікроконтролера тільки цифрових кодів. Інші ж, такі як L6506, L298 фірми SGS-Thomson, вимагають зовнішніх опорних напруг синусоїдальної форми, які повинен формувати мікроконтролер за допомогою ЦАП.

Потрібно сказати, що занадто велика кількість дискретних відліків синуса не приводить до підвищення точності позиціонування, тому що починає домінувати помилка, пов'язана з неідеальністю геометрії полюсів двигуна. Тим більше, у цьому випадку відліки повинні визначатися з великою частотою, що є проблемою при їхньому програмній реалізації.

При роботі на більших швидкостях роздільну здатність ЦАП можна зменшити. Крім того, при дуже великих швидкостях взагалі рекомендується працювати у звичайному повнокроковому режимі, тому що режим керування гармонічним сигналом втрачає свої переваги.

Відбувається це з тієї причини, що обмотки двигуна являють собою індуктивність, відповідно будь-яка конкретна схема драйвера з конкретною напругою живлення забезпечує цілком певну максимальну швидкість наростання струму. Тому при підвищенні частоти форма струмів починає відхилятися від синусоїдальної й на дуже великих частотах стає трикутною.

Залежність моменту від швидкості, вплив навантаження

Момент, створюваний кроковим двигуном, залежить від декількох факторів:

- швидкості;
- величини струму в обмотках;
- схеми драйвера.

На рис. 1а показана залежність моменту від кута повороту ротора.

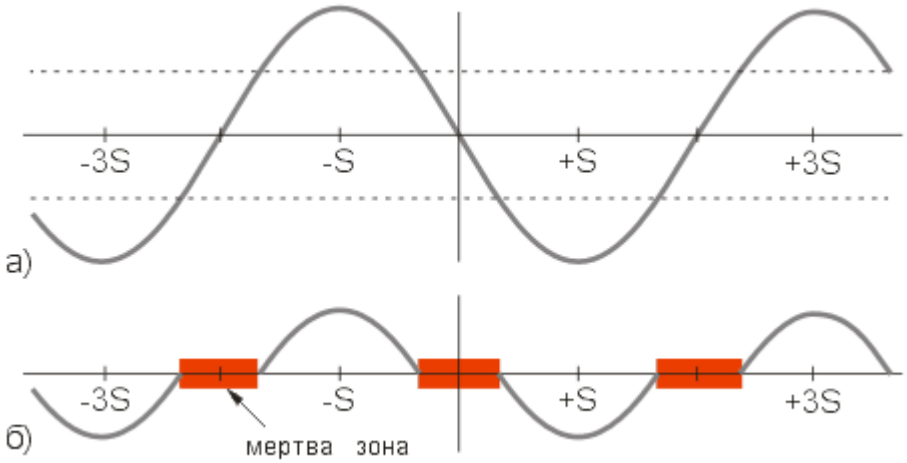


Рис. 1 - Виникнення мертвих зон у результаті дії тертя

В ідеального крокового двигуна ця залежність синусоїдальна. Точки S є положеннями рівноваги ротора для невантаженого двигуна й відповідають декільком послідовним крокам. Якщо до вала двигуна прикласти зовнішній момент, менший моменту утримання, то кутове положення ротора зміниться на деякий кут Ф:

$$\Phi = (N/(2*pi))*\sin(Ta/Th),$$

де Ф - кутовий зсув,;

N - кількість кроків двигуна на оберт;

Ta - зовнішній прикладений момент;

Th - момент утримання.

Кутовий зсув Ф є похибкою позиціонування навантаженого двигуна. Якщо до валу двигуна прикласти момент, що перевищує момент утримання, то під дією цього моменту вал провернеться. У такому режимі положення ротора є неконтрольованим.

На практиці завжди є прикладений до двигуна зовнішній момент, хоча б тому, що двигуну доводиться переборювати тертя. Сили тертя можуть бути розділені на дві категорії: статичне тертя або тертя спокою, для подолання якого потрібен постійний момент і динамічне тертя або в’язке тертя, що залежить від швидкості.

Розглянемо статичне тертя. Припустимо, що для його подолання потрібен момент у половину від максимального. На рис. 1а штриховими лініями показаний момент тертя. Таким чином, для обертання ротора залишається тільки момент, що лежить на графіку за межами штрихових ліній. Звідси випливають два виводи: тертя знижує момент на валу двигуна й з'являються мертві зони навколо кожного положення рівноваги ротора (рис. 1б):

$$d = 2 (S / (pi/2)) \arcsin(T f /T h) = (S / (pi/4)) \arcsin(T f / Th),$$

де d - ширина мертвої зони в радіанах;

S - кут кроку в радіанах;

Tf - момент тертя;

Th - момент утримання.

Мертві зони обмежують точність позиціонування. Наприклад, наявність статичного тертя в половину від максимального моменту двигуна із кроком 90 град. викличе наявність мертвих зон в 60 град. Це означає, що крок двигуна може коливатися від 30 до 150 град., залежно від того, у якій точці мертвої зони зупиниться ротор після чергового кроку.

Наявність мертвих зон є дуже важливим для мікрокрокового режиму. Якщо, наприклад, є мертві зони величиною d, то мікрокрок величиною менш d взагалі не зрушить ротор з місця. Тому для систем з використанням мікрокрокових режимів дуже важливо мінімізувати тертя спокою.

Контрольні питання

1. Які фактори визначають момент крокового двигуна?
2. Поясніть, яким чином мертві зони обмежують точність позиціонування?
3. Яким чином забезпечується синусоїдальний струм фаз?
4. Поясніть джерела похибок позиціонування.
5. Поясніть вплив статичного тертя на точність позиціонування двигуна.

Лекція 13

ФОРМУВАННЯ ПЕРЕХІДНИХ ПРОЦЕСІВ

Коли двигун працює під навантаженням, завжди існує деяке зрушення між кутовим положенням ротора й орієнтацією магнітного поля статора. Особливо несприятливої є ситуація, коли двигун починає гальмування і знак моменту навантаження змінюється на протилежний.

Потрібно відзначити, що запізнення або випередження відноситься лише до положення, але не до швидкості. У будь-якому разі, якщо синхронність роботи двигуна збережена, це запізнення або випередження не може перевищувати величини двох повних кроків. Це досить приємний факт.

Щораз, коли кроковий двигун здійснює крок, ротор повертається на S радіан. При цьому мінімальний момент має місце, коли ротор перебуває рівно між сусідніми положеннями рівноваги (рис. 1).

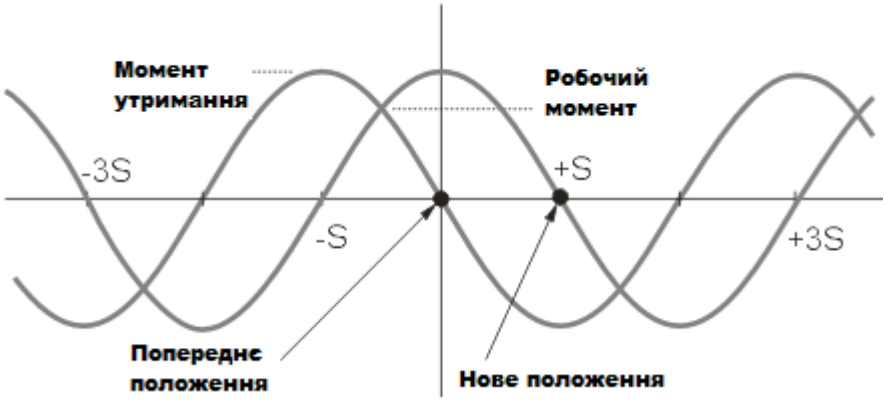


Рис. 1- Момент утримання й робочий момент крокового двигуна

Цей момент називають робочим моментом. Робочий момент визначає, який найбільший момент навантаження може витримувати двигун при обертанні з малою швидкістю.

При синусоїдальній залежності моменту від кута повороту ротора, цей момент Tr = Th/1,41. Якщо двигун робить крок із двома підключеними обмотками, то робочий момент дорівнює моменту утримання для однієї підключеної обмотки.

Параметри приводу на основі крокового двигуна сильно залежать від характеристик навантаження. Крім тертя, реальне навантаження має інерцію. Інерція перешкоджає зміні швидкості. Інерційне навантаження потребує від двигуна більших моментів на розгоні й гальмуванні, обмежуючи в такий спосіб максимальне прискорення. З іншого боку, збільшення моменту інерції навантаження збільшує стабільність швидкості.

Такий параметр крокового двигуна, як залежність моменту від швидкості є найважливішим при виборі типу двигуна, виборі методу керування фазами й виборі схеми драйвера. При конструюванні високошвидкісних драйверів крокових двигунів потрібно враховувати, що обмотки двигуна являють собою індуктивність. Ця індуктивність визначає час наростання й спаду струму. Тому якщо до обмотки прикладена напруга прямокутної форми, форма струму не буде прямокутною.

При низьких швидкостях (рис. 2а) час наростання й спаду струму не здатний сильно вплинути на момент, однак на високих швидкостях момент двигуна зменшується. Пов'язано це з тим, що на високих швидкостях струм в обмотках двигуна не встигає досягти номінального значення (рис. 2б).

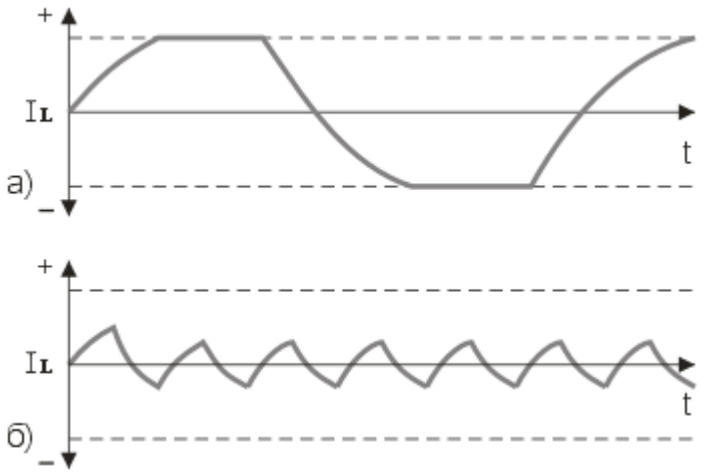


Рис. 2. Форма струму в обмотках двигуна при різних швидкостях

Для того, щоб момент падав якнайменше, необхідно забезпечити високу швидкість наростання струму в обмотках двигуна, що досягається застосуванням спеціальних схем для їхнього живлення.

Поводження моменту при збільшенні частоти комутації фаз приблизно таке: починаючи з деякої частоти зрізу момент монотонно падає. Звичайно для крокового двигуна приводяться дві криві залежності моменту від швидкості (рис. 3).

Рис. 3. Залежність моменту від швидкості

Внутрішня крива (крива старту, або pull-in curve) показує, при якому максимальному моменті тертя для даної швидкості кроковий двигун здатний рушити. Ця крива перетинає вісь швидкостей у точці, яка називається максимальною частотою старту або частотою прийнятності. Вона визначає максимальну швидкість, на якій ненавантажений двигун може рушити. На практиці ця величина лежить у межах 200 - 500 повних кроків за секунду.

Момент інерції навантаження сильно впливає на вид внутрішньої кривої. Більший момент інерції відповідає меншій області під кривою. Ця область називається областю старту. Зовнішня крива (крива розгону, або pull-out curve) показує, при якому максимальному моменті тертя для даної швидкості кроковий двигун здатний підтримувати обертання без пропуску кроків. Ця крива перетинає вісь швидкостей у точці, яка називається максимальною частотою розгону. Вона показує максимальну швидкість для даного двигуна без навантаження. При вимірі максимальної швидкості потрібно мати на увазі, що через явище резонансу момент дорівнює нулю ще й на резонансній частоті. Область, що лежить між кривими, називається областю розгону. Потрібно відзначити, що схема драйвера в значній мірі впливає на вид цих характеристик.

Розгін та гальмування двигуна

Для того, щоб працювати на великій швидкості з області розгону (рис. 3), необхідно стартувати на низької швидкості з області старту, а потім виконати розгін. При зупинці потрібно діяти у зворотному порядку: спочатку виконати гальмування, і тільки ввійшовши в область старту можна припинити подачу керуючих імпульсів. У противному випадку відбудеться втрата синхронності й положення ротора буде загублено.

Використання розгону й гальмування дозволяє досягти значно більших швидкостей - в індустріальних застосуваннях використовуються

швидкості до 10000 повних кроків у секунду. Необхідно відзначити, що безперервна робота крокового двигуна на високій швидкості не завжди припустима через нагрівання ротора. Однак висока швидкість короткочасно може бути використана при здійсненні позиціонування.

При розгоні двигун проходить ряд швидкостей, при цьому на одній зі швидкостей можна зіштовхнутися з неприємним явищем резонансу. Для нормального розгону бажано мати навантаження, момент інерції якої як мінімум дорівнює моменту інерції ротора. На ненавантаженому двигуні явище резонансу проявляється найбільше сильно. Докладніше методи боротьби із цим явищем будуть розглянуті далі.

При здійсненні розгону або гальмування важливо правильно вибрати закон зміни швидкості й максимальне прискорення. Прискорення повинне бути тим менше, чим вище момент інерції навантаження. Критерій правильного вибору режиму розгону - це здійснення розгону до потрібної швидкості для конкретного навантаження за мінімальний час. На практиці найчастіше застосовують розгін і гальмування з постійним прискоренням.

Реалізація закону, по якому буде виробляється прискорення або гальмування двигуна, звичайно виробляється програмно керуючим мікроконтролером, тому що саме мікроконтролер звичайно є джерелом тактової частоти для драйвера крокового двигуна. Хоча раніше для цих цілей застосовувалися керовані напругою генератори або програмовані дільники частоти.

Для генерації тактової частоти зручно використати апаратний таймер, що є в складі практично будь-якого мікроконтролера. Коли двигун обертається з постійною швидкістю, досить завантажити в таймер постійне значення періоду повторення кроків (тривалість кроку).

Якщо ж двигун розганяє або гальмується, цей період міняється з кожним новим кроком. При розгоні або гальмуванні з постійним прискоренням частота повторення кроків повинна змінюватися лінійно,

відповідно значення періоду, яких необхідно завантажувати в таймер, повинне мінятися за гіперболічним законом.

Для найбільш загального випадку потрібно знати залежність тривалості кроку від поточної швидкості. Кількість кроків, що здійснює двигун при розгоні за час t дорівнює:

$$N = 1/2At2+Vt,$$

де N - число кроків;
t – час;
V - швидкість, виражена в кроках в одиницю часу;
A - прискорення, виражене в кроках, ділених на час у квадраті.

Для одного кроку N = 1, тоді тривалість кроку

$$t1 = T = (-V+(V2+2A)0.5)/A.$$

В результаті здійснення кроку швидкість стає рівною

$$Vnew = (V2+2A)0.5.$$

Обчислення по наведених формулах досить трудомісткі й вимагають значних витрат процесорного часу. У той же час, вони дозволяють змінювати значення прискорення в довільний момент. Розрахунки можна істотно спростити, якщо зажадати сталості прискорення під час розгону й гальмування.

У цьому випадку можна записати залежність тривалості кроку від часу розгону:

$$V = V0+At,$$

де V - поточна швидкість;

V0 - початкова швидкість (мінімальна швидкість, з якої починається розгін);

A – прискорення.

$1/T = 1/T0+At,$

де T - тривалість кроку;

T0 - початкова тривалість кроку;

t - поточний час.

Звідки

$T = T0/(1+T0At).$

Обчислення по цій формулі здійснити значно простіше, однак для того, щоб поміняти значення прискорення, потрібно зупинити двигун.

Контрольні питання

1. Поясніть поняття робочого моменту двигуна.
2. Як виконується розгін та гальмування двигуна?
3. Яке навантаження повинен мати двигун для зменшення впливу резонансу?
4. Поясніть поняття максимальної швидкості старту.
5. У чому полягає критерій правильного вибору режиму розгону?

Лекція 14

РЕЗОНАНСНІ ЯВИЩА В КРОКОВОМУ ПРИВОДІ

Крокови двигунам властивий небажаний ефект, називаний резонансом. Ефект проявляється у вигляді раптового падіння моменту на деяких швидкостях. Це може привести до пропуску кроків і втраті синхронності. Ефект проявляється ситуації, коли частота кроків збігається із власною резонансною частотою ротора двигуна.

Коли двигун робить крок, ротор не відразу встановлюється в нову позицію, а робить загасаючі коливання. Справа в тому, що систему ротор - магнітне поле - статор можна розглядати як пружинний маятник, частота коливань якого залежить від моменту інерції ротора (плюс навантаження) і величини магнітного поля. Через складну конфігурацію магнітного поля, резонансна частота ротора залежить від амплітуди коливань.

При зменшенні амплітуди частота росте, наближаючись до малоампітудної частоти, яка більш просто обчислюється кількісно. Ця частота залежить від кута кроку та від відношення моменту утримання до моменту інерції ротора. Більший момент утримання й менший момент інерції приводять до збільшення резонансної частоти.

Резонансна частота обчислюється по формулі:

F0 = (N*TH/(JR+JL))0.5/4*pi,

де

F0 - резонансна частота;

N - число повних кроків на оберт;

TH - момент утримання для використовуваного способу керування й струму фаз;

JR - момент інерції ротора;

JL - момент інерції навантаження.

Необхідно відмітити, що резонансну частоту визначає момент інерції власне ротора двигуна плюс момент інерції навантаження, під'єднаного до вала двигуна. Тому резонансна частота ротора ненавантаженого двигуна, що іноді приводиться серед параметрів, має малу практичну цінність, оскільки будь-яке навантаження, приєднане до двигуна, змінить цю частоту.

На практиці ефект резонансу приводить до труднощів при роботі на частоті, близької до резонансної. Момент на частоті резонансу дорівнює нулю й без вживання спеціальних заходів кроковий двигун не може при розгоні пройти резонансну частоту. У кожному разі, явище резонансу здатне істотно погіршити точністні характеристики приводу.

У системах з низьким демпфуванням існує небезпека втрати кроків або підвищення шуму, коли двигун працює поблизу резонансної частоти. У деяких випадках проблеми можуть виникати й на гармоніках частоти основного резонансу.

Коли робота відбувається в не мікрокроковому режимі, основною причиною появи коливань є переривчасте обертання ротора. При здійсненні кроку ротору поштовхом передається деяка енергія. Цей поштовх збуджує коливання. Енергія, що передається ротору в напівкроковому режимі, становить близько 30% від енергії повного кроку. Тому в напівкроковому режимі амплітуда коливань істотно менше. У мікрокроковому режимі із кроком 1/32 основного при кожному мікрокроці ротору передається всього близько 0.1% від енергії повного кроку. Тому в мікрокроковому режимі явище резонансу практично непомітно.

Для боротьби з резонансом можна використати різні методи. Наприклад, застосування еластичних матеріалів при виконанні механічних муфт зв'язку з навантаженням. Еластичний матеріал сприяє поглинанню

енергії в резонансній системі, що приводить до загасання паразитних коливань.

Іншим способом є застосування в’язкого тертя. Випускаються спеціальні демпфери, де усередині порожнього циліндра, заповненого в'язким кремнійорганічним змащенням, може обертатися металевий диск. При обертанні цієї системи із прискоренням диск створює в’язке тертя, що ефективно демпфірує систему.

Існують електричні методи боротьби з резонансом. Коливання ротору при резонансі приводить до виникнення в обмотках статора ЕРС. Якщо замкнути накоротко обмотки, які на даному кроці не використовуються, це приведе до демпфірування резонансу.

І, нарешті, існують методи боротьби з резонансом на рівні алгоритму роботи драйвера. Наприклад, можна використати той факт, що при роботі із двома включеними фазами резонансна частота приблизно на 20% вище, ніж з однією включеною фазою. Якщо резонансна частота точно відома, то її можна проходити, міняючи режим роботи.

Якщо це можливо, при старті й зупинці потрібно використовувати частоти вище резонансної. Збільшення моменту інерції системи ротор-навантаження зменшує резонансну частоту.

Тим не менше, найефективнішою мірою для боротьби з резонансом важається застосування мікрокрокового режиму.

Контрольні питання

1. Поясніть явище резонансу у кроковому приводі.
2. Поясніть методи боротьби з резонансом у кроковому приводу.
3. Що є причиною появи коливань у кроковому приводі?
4. Чому дорівнює момент двигуна на частоті резонансу?
5. Які фактори впливають на резонансну частоту крокового двигуна?

Лекція 15

СПОСОБИ ЖИВЛЕННЯ КРОКОВИХ ДВИГУНІВ

Для живлення звичайного двигуна постійного струму потрібно лише джерело постійної напруги, а необхідні комутації обмоток виконуються колектором. Із кроковим двигуном усе складніше. Всі комутації повинен виконувати зовнішній контролер. На сьогодні в більшості випадків для керування кроковими двигунами застосовуються мікроконтролери.

У найпростішому випадку для керування кроковим двигуном у полношаговом режимі потрібні всього два сигнали, зміщені по фазі на 90 градусів. Напрям обертання залежить від того, яка фаза випереджає. Швидкість визначається частотою проходження імпульсів.

У напівкроковому режимі цей процес трохи складніший і потрібно вже мінімум 4 сигнали. Всі сигнали керування кроковим двигуном можна сформувати програмно, однак це приводить до більш суттєвого завантаження мікроконтролера. Тому частіше застосовують спеціальні мікросхеми драйверів крокового двигуна, які зменшують кількість необхідних від процесора динамічних сигналів. Типово ці мікросхеми потребують зовнішнього джерела тактової частоти, що є частотою повторення кроків, а також статичного сигналу, що задає напрямок. Іноді ще потрібен сигнал вмикання напівкрокового режиму.

Для мікросхем драйверів, які працюють у мікрокроковому режимі, потрібна більша кількість сигналів. Розповсюдженим є випадок, коли необхідні послідовності сигналів керування фазами формуються за допомогою однієї мікросхеми, а необхідні струми фаз забезпечує інша мікросхема. Хоча останнім часом з'являється усе більше драйверів, що реалізують всі функції в одній мікросхемі.

Потужність, яку повинен забезпечувати драйвер, залежить від розмірів двигуна й становить частки Вата для маленьких двигунів і до 10-20 Ватів для більших двигунів. Максимальний рівень потужності розсіювання, обмежений нагріванням двигуна. Максимальна робоча температура звичайно вказується виробником, але можна приблизно вважати, що нормальною є температура корпусу 90 градусів.

Тому при конструюванні пристроїв із кроковими двигунами, що безупинно працюють на максимальному струмі, необхідно приймати міри, що виключають доторкання обслуговуючим персоналом до корпусу двигуна. У окремих випадках можливе застосування охолоджувального радіатора. Іноді це дозволяє застосувати двигун менших розмірів і домогтися кращого відношення потужність/вартість.

Для заданого розміру крокового двигуна місце, займане обмотками, обмежене. Тому дуже важливо конструювати драйвер так, щоб для даних параметрів обмоток забезпечити найкращу ефективність.

Схема драйвера повинна виконувати три головних завдання:

- мати можливість включати й виключати струм в обмотках, а також міняти його напрямок;
- підтримувати задане значення струму;
- забезпечувати як можна більш швидке наростання й спад струму в обмотках для гарних швидкісних характеристик

Способи зміни напрямку струму

При роботі крокового двигуна потрібна зміна напрямку магнітного поля незалежно для кожної фази. Зміну напрямку магнітного поля може бути виконано різними способами.

В уніполярних двигунах обмотки мають відвід від середини або є дві окремі обмотки для кожної фази. Напрямок магнітного поля міняється

шляхом перемикання половинок обмоток або цілих обмоток. У цьому випадку потрібні тільки два простих ключі A і B для кожної фази (рис. 1).

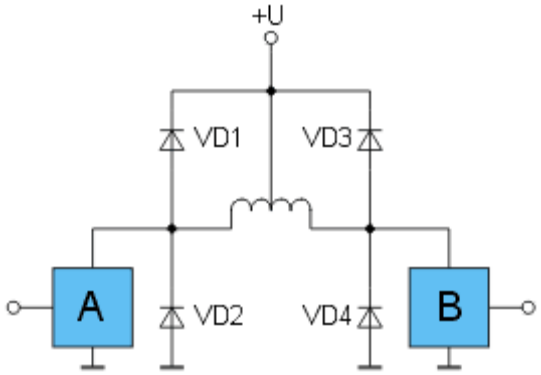


Рис. 1. Живлення обмотки уніполярного двигуна

У біполярних двигунах напрямок міняється шляхом зміни полярності напруги на виводах обмоток. Для такої зміни потрібен повний Н-міст (рис. 2). Керування ключами в обох випадках повинне здійснюватися логічною схемою, що реалізує потрібний алгоритм роботи. Передбачається, що джерело живлення схем має номінальну для обмоток двигуна напругу.

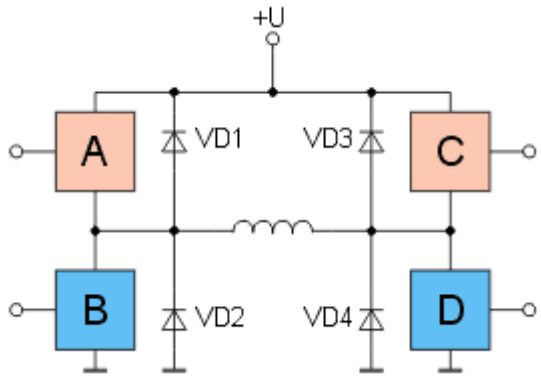


Рис. 2. Живлення обмотки біполярного двигуна

Це найпростіший спосіб керування струмом обмоток, і як буде показано надалі, він істотно обмежує можливості двигуна. Потрібно

відзначити, що при роздільному керуванні транзисторами Н-мосту можливі аварійні ситуації, коли джерело живлення виявиться замкненим накоротко ключами одного плеча моста. Тому логічна схема керування повинна бути побудована таким чином, щоб виключити цю ситуацію навіть у випадку збоїв керуючого мікроконтролера.

Обмотки двигуна являють собою індуктивність, а це значить, що струм не може нескінченно швидко наростати або нескінченно швидко спадати без залучення нескінченної різниці потенціалів. При підключенні обмотки до джерела живлення струм буде з деякою швидкістю наростати, а при відключенні обмотки відбудеться викид напруги.

Цей викид здатний ушкодити ключі, у якості яких використаються біполярні або польові транзистори. Для обмеження цього викиду встановлюють спеціальні захисні ланцюжки. На схемах рис. 1 і 2 ці ланцюжки утворені діодами, значно рідше застосовують конденсатори або їхню комбінацію з діодами. Застосування конденсаторів викликає появу електричного резонансу, що може викликати збільшення моменту на деякій швидкості.

У схемі рис. 1 потрібно було 4 діоди з тієї причини, що половинки обмоток уніполярного двигуна розташовані на загальному сердечнику й магнітно сильно зв'язані між собою. Вони працюють як автотрансформатор і викиди напруги виникають на виводах обох обмоток. Якщо ж в якості ключів застосовані MOS-транзистори, то досить тільки двох зовнішніх діодів, тому що в цих транзисторах всередині вже є діоди.

В інтегральних мікросхемах, що містять потужні вихідні каскади з відкритим колектором, також часто є такі діоди. Крім того, деякі мікросхеми, такі як ULN2003, ULN2803 і подібні мають усередині обидва захисних діоди для кожного транзистора. Потрібно відзначити, що у випадку застосування швидкодіючих ключів потрібні порівнянні по

швидкодії діоди. У випадку застосування повільних діодів потрібно їхнє шунтування невеликими конденсаторами.

Стабілізація струму

Для регулювання моменту потрібно регулювати величину струму в обмотках. У будь-якому разі, струм повинен бути обмежений, щоб не перевищити припустиму потужність розсіювання на омічному опорі обмоток. Більш того, у напівкроковому режимі ще потрібно в певні моменти забезпечувати нульове значення струму в обмотках, а в мікрокроковому режимі взагалі потрібне завдання різних значень струму.

Для кожного двигуна виробником вказується номінальна робоча напруга обмоток. Тому найпростіший спосіб живлення обмоток - це використання джерела постійної напруги. У цьому випадку струм обмежений омічним опором обмоток і напругою джерела живлення (рис. 3а), тому такий спосіб живлення називають L/R-живленням. Струм в обмотці наростає за експонентним законом зі швидкістю, обумовленою індуктивністю, активним опором обмотки й прикладеною напругою. При підвищенні частоти струм не досягає номінального значення й момент падає. Тому такий спосіб живлення придатний тільки при роботі на малих швидкостях і використовується на практиці тільки для малопотужних двигунів.

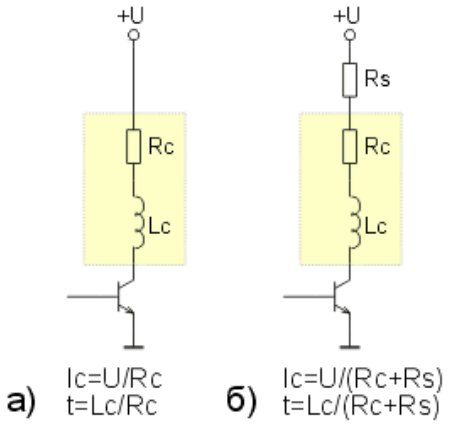


Рис. 3. Живлення обмотки номінальною напругою (а)
і використання обмежувального резистора (б).

При роботі на більших швидкостях потрібно збільшувати швидкість наростання струму в обмотках, що можливо шляхом підвищення напруги джерела живлення. При цьому максимальний струм обмотки повинен бути обмежений за допомогою додаткового резистора. Наприклад, якщо використовується напруга живлення в 5 разів більша номінальної, то потрібно такий додатковий резистор, щоб загальний опір складав 5R, де R - омічний опір обмотки (L/5R-живлення).

Цей спосіб живлення забезпечує більш швидке наростання струму і як наслідок, більший момент (рис. 3б). Однак він має істотний недолік: на резисторі розсіюється додаткова потужність. Більші габарити потужних резисторів, необхідність відводу тепла й підвищена необхідна потужність джерела живлення - все це робить такий метод неефективним і обмежує область його застосування невеликими двигунами потужністю 1 - 2 Вт.

Потрібно відмітити, що до початку 80-х років минулого століття параметри крокових двигунів, що приводять виробниками, встановлювалися саме для такого способу живлення.

Ще більш швидке наростання струму можна одержати, якщо використати для живлення двигуна генератор струму. Наростання струму буде відбуватися лінійно, це дозволить швидше досягати номінального значення струму. Тим більше, що пара потужних резисторів може коштувати дорожче, ніж пара потужних транзисторів разом з радіаторами. Але як і в попередньому випадку, генератор струму буде розсіювати додаткову потужність, що робить цю схему живлення неефективної.

Існує ще одне рішення, що забезпечує високу швидкість наростання струми й низьку потужність втрат. Засновано воно на застосуванні двох джерел живлення.

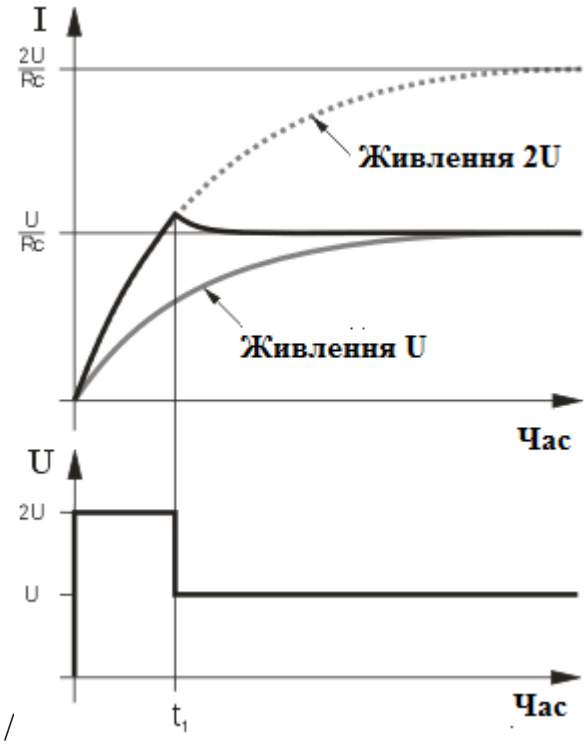


Рис. 4. Живлення обмотки двигуна ступінчастою напругою

На початку кожного кроку короткочасно обмотки підключаються до більше високовольтного джерела, що забезпечує швидке наростання струму (рис. 4). Потім напруга живлення обмоток зменшується (момент часу t1 на рис. 4).

Недоліком цього методу є необхідність двох ключів, двох джерел живлення й більш складної схеми керування. У системах, де такі джерела уже є, метод може виявитися досить дешевим.

Додаткова складність полягає у неможливості визначення моменту часу t1 для загального випадку. Для двигуна з меншою індуктивністю обмоток швидкість наростання струму вище й при фіксованому t1 середній струм може виявитися вище номінального, що може привести до перегріву двигуна.

Контрольні питання

1. Поясніть схеми живлення уніполярних та біполярних двигунів.
2. Яким вимогам має відповідати схема драйвера?
3. Поясніть способи підвищення швидкості наростання струму в обмотках крокового двигуна.
4. Поясніть спосіб живлення двигуна ступінчатою напругою.
5. Поясніть способи та схеми зміни напряму струму в омотках крокових двигунів.

Лекція 16

ПРАКТИЧНА РЕАЛІЗАЦІЯ ДРАЙВЕРІВ КРОКОВИХ ДВИГУНІВ

При ключовому регулюванні величина пульсацій струму залежить від швидкості його спаду. Тут можливі різні варіанти.

Якщо забезпечити замикання обмотки діодом, буде реалізований повільний спад струму. Це приводить до зменшення амплітуди пульсацій струму, що є досить бажаним, особливо при роботі двигуна в мікрокроковому режимі. Для даного рівня пульсацій повільний спад струму дозволяє працювати на більше низьких частотах ШІМ, що зменшує нагрівання двигуна.

Із цих причин повільний спад струму широко використовується. Однак існує кілька причин, з яких повільне спадання струму не завжди є оптимальним:

- по-перше, через вплив зворотної ЕРС, через малу напругу на обмотці під час спаду струму, реальний середній струм обмотки може виявитися завищеним;
- по-друге, коли потрібно різко зменшити струм фази (наприклад, у напівкроковому режимі), повільний спад не дозволить зробити це швидко;
- по-третє, коли потрібно встановити дуже низьке значення струму фази, регулювання може порушиться через існування обмеження на мінімальний час увімкненого стану ключів.

Висока швидкість спаду струму, що реалізується шляхом замикання обмотки на джерело живлення, приводить до підвищених пульсацій. Разом з тим, усуваються недоліки, властиві повільному спаду струму. Однак при цьому точність підтримки середнього струму менше, а також більші втрати.

Найбільш просунуті (advanced) мікросхеми драйверів мають можливість регулювати швидкість спаду струму.

Практична реалізація драйверів

Драйвер крокового двигуна повинен вирішувати два основні завдання: -

- формування необхідних часових послідовностей сигналів;
- забезпечення необхідного струму в обмотках.

В інтегральних реалізаціях, як правило, ці завдання виконуються різними мікросхемами. Прикладом може служити комплект мікросхем L297 і L298 фірми SGS-Thomson. Мікросхема L297 містить логічні схеми для формування часових послідовностей, а L298 являє собою потужний здвоєний Н-міст.

На жаль, існує деяка плутанина в термінології щодо подібних мікросхем. Поняття «драйвер» часто застосовують до багатьом мікросхемам, навіть якщо їхні функції сильно розрізняються. Іноді мікросхеми логіки називають «трансляторами». Далі будемо використовувати наступну термінологію: «контролер» - мікросхема, відповідальна за формування часових послідовностей; «драйвер» - потужна схема живлення обмоток двигуна.

Однак терміни «драйвер» і «контролер» можуть також позначати закінчений пристрій керування кроковим двигуном. Необхідно відзначити, що останнім часом всі частіше контролер і драйвер поєднуються в одній мікросхемі.

На практиці можна обійтися й без спеціалізованих мікросхем. Наприклад, всі функції контролера можна реалізувати програмно, а в якості драйвера застосувати набір дискретних транзисторів. Однак при цьому мікроконтролер буде сильно завантажений, а реалізація драйвера може

вийде більш громіздкою. Незважаючи на це, у деяких випадках таке рішення буде економічно вигідним.

Найпростіший драйвер потрібен для керування обмотками уніполярного двигуна. Для цього підходять найпростіші ключі, у якості яких можуть бути використані біполярні або польові транзистори. Досить ефективні потужні MOS-транзистори, керовані логічним рівнем, такі як IRLZ34, IRLZ44, IRL540. У них опір у відкритому стані менш 0.1Ом і припустимий струм порядку 30А.

Існують також спеціальні мікросхеми, які містять всередині кілька потужних транзисторних ключів. Прикладом може служити мікросхема ULN2003 фірми Allegro), що містить 7 ключів з максимальним струмом 0.5 А. Принципова схема одного осередку цієї мікросхеми наведена на рис. 1.

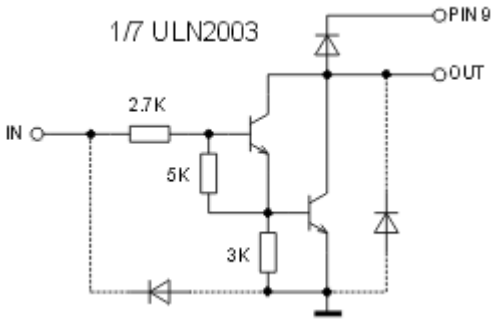


Рис. 1. Принципова схема одного осередку мікросхеми ULN2003

Аналогічні мікросхеми випускаються багатьма фірмами. Необхідно відзначити, що ці мікросхеми придатні не тільки для живлення обмоток крокових двигунів, але й для живлення будь-яких інших навантажень.

Крім простих мікросхем драйверів існують і більше складні мікросхеми, що мають вбудований контролер, ШІМ-регулювання струму й навіть ЦАП для мікрокрокового режиму.

Як ми вже відзначали раніше, для керування біполярними двигунами потрібні більше складні схеми, такі як Н-мости. Такі схеми теж можна реалізувати на дискретних елементах, хоча останнім часом всі частіше вони реалізуються за допомогою інтегральних схемах. Приклад дискретної реалізації показано на рис. 2.

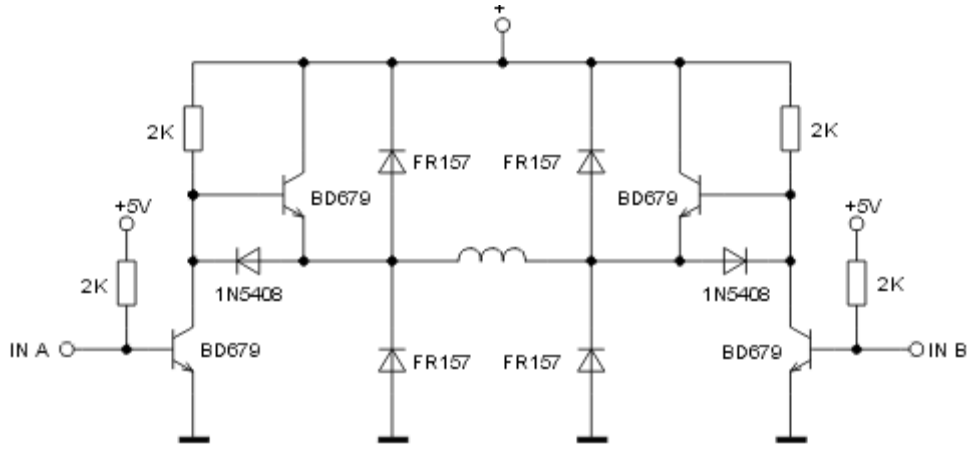


Рис. 2. Реалізація мостового драйвера на дискретних компонентах

Такий Н-міст управляється за допомогою двох сигналів, тому він не дозволяє забезпечити всіх можливих комбінацій. Обмотка підключена до напруги живлення, коли рівні на входах різні й замкнена накоротко, коли рівні однакові. Це дозволяє реалізувати тільки режим повільного спаду струму (динамічне гальмування). Мостові драйвери в інтегральному виконанні випускаються багатьма фірмами. Прикладом можуть служити L293 і L298 фірми SGS-Thomson.

Як прості ключі, так і Н-мости можуть становити частину ключового стабілізатора струму. Схема керування ключами може бути виконана на дискретних компонентах або у вигляді спеціалізованої мікросхеми. Досить популярним пристроєм, що реалізує ШІМ-стабілізацію струму, є мікросхема L297 фірми SGS-Thomson. Разом з мікросхемою мостового драйвера L293 або L298 вони утворюють закінчену систему керування для крокового двигуна (рим. 3).

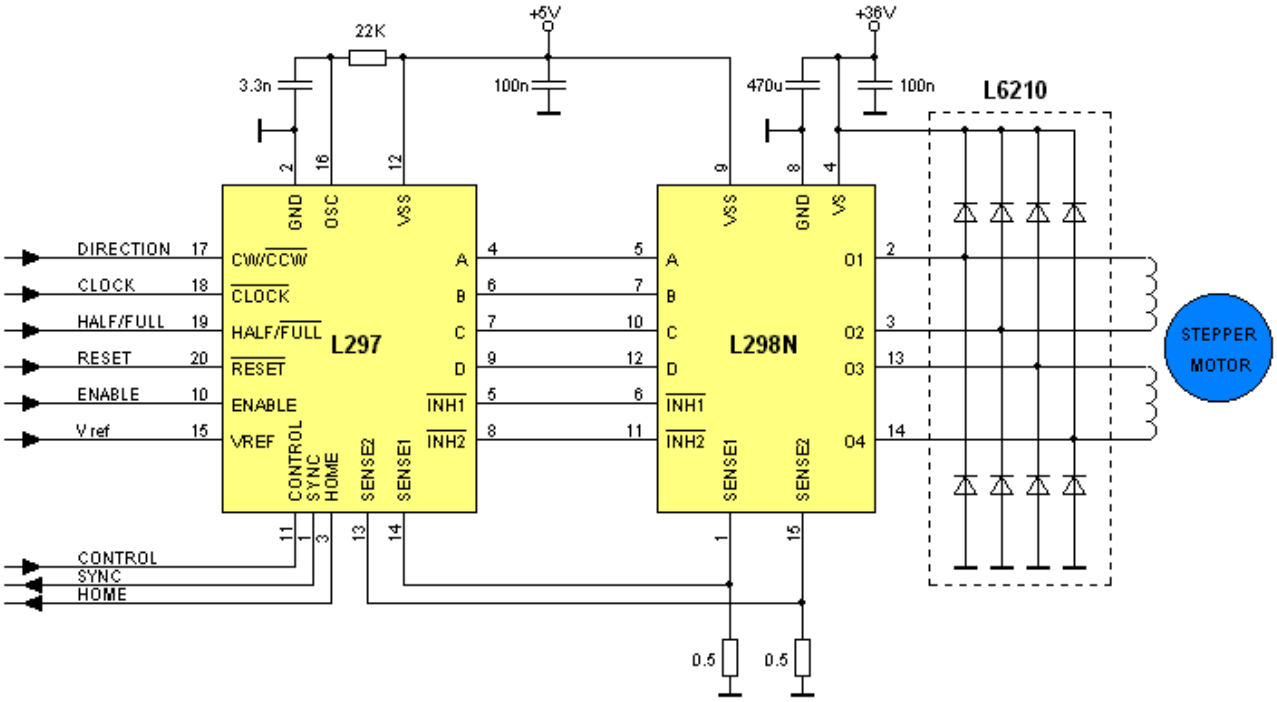


Рис. 3. Типова схема включення мікросхем L297 і L298N

Мікросхема L297 значно розвантажує керуючий мікроконтролер, тому що від нього потрібні тільки тактова частота CLOCK (частота повторення кроків) та кілька статичних сигналів: DIRECTION - напрямок (сигнал внутрішньо синхронізований, перемикати можна в будь-який момент), HALF/FULL - напівкроковий/повнокроковий режим, RESET - установлює фази у вихідний стан (ABCD = 0101), ENABLE - дозвіл роботи мікросхеми, V ref - опорна напруга, що задає пікову величину струму при ШІМ-регулюванні.

Крім того, є кілька додаткових сигналів. Сигнал CONTROL задає режим роботи ШІМ-регулятора. При його низькому рівні ШІМ-регулювання відбувається по виходах INH1, INH2, а при високому - по виходах ABCD. SYNC - вихід внутрішнього тактового генератора ШІМ. Він служить для синхронізації роботи декількох мікросхем, а також може бути використаний як вхід при тактуванні від зовнішнього генератора. HOME - сигнал початкового положення (ABCD = 0101). Він

використовується для синхронізації перемикання режимів HALF/FULL. В залежності від моменту переходу в повнокроковий режим мікросхема може працювати в режимі з однією включеною фазою або із двома включеними фазами.

Ключове регулювання реалізують і багато інших мікросхем. Деякі мікросхеми мають ті або інші особливості, наприклад драйвер LMD18T245 фірми National Semiconductor не вимагає застосування зовнішнього датчика струму, тому що він реалізований всередині на основі одного осередку ключового MOS-транзистора.

Деякі мікросхеми призначені спеціально для роботи в мікрокроковому режимі. Прикладом може служити мікросхема A3955 фірми Allegro. Вона має вбудований 3-бітний нелінійний ЦАП для завдання зміни струму фази по синусоїдальному закону.

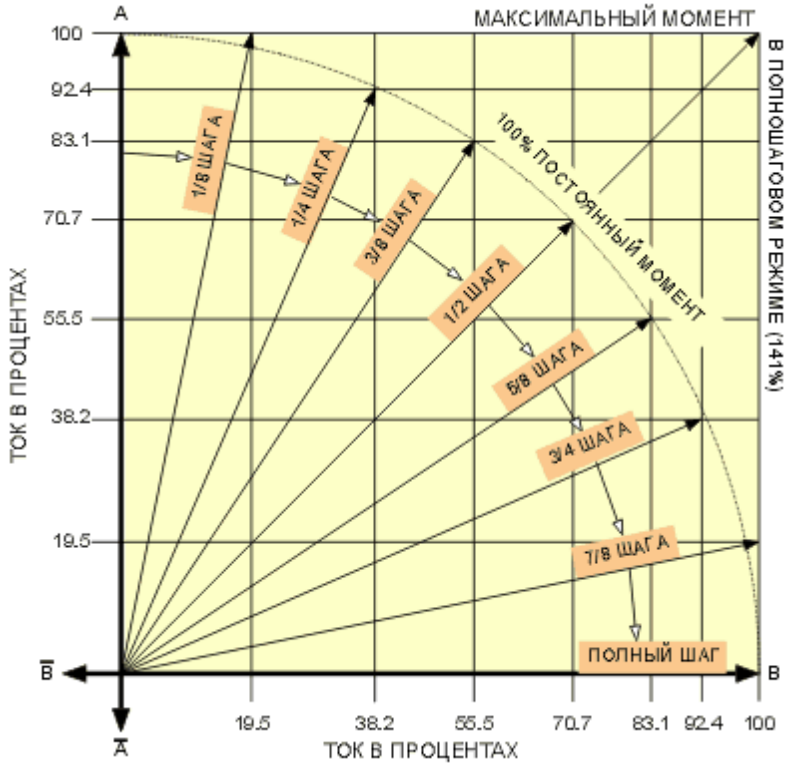


Рис. 4. Струм і вектор зсуву ротора

Зсув ротора залежно від струмів фаз, які сформовані цим 3-бітним Цапом, показане на рис. 4.

Мікросхема A3972 має вбудований 6-бітний лінійний ЦАП.

Контрольні питання

1. Поясніть основні завдання драйвера крокового двигуна.
2. Яким чином реалізується режим повільного спаду струму?
3. Поясніть поняття драйвера та контролера.
4. Як реалізовані датчики струму в обмотках двигуна?
5. Поясніть реалізацію драйвера на дискретних елементах.

Лекція 17

СИСТЕМА КЕРУВАННЯ КРОКОВИМ ДВИГУНОМ НА ОСНОВІ МІКРОКОНТРОЛЕРА

Максимальний момент і потужність, що може забезпечити на валу кроковий двигун, залежить від розмірів двигуна, умов охолодження, режиму роботи (відносини робота/пауза), від параметрів обмоток двигуна й від типу застосовуваного драйвера. Тип застосовуваного драйвера також сильно впливає на потужність на валу двигуна. При однаковій розсіюваній потужності, драйвер з імпульсною стабілізацією струму забезпечує виграш у моменті на деяких швидкостях до 5 - 6 разів, у порівнянні з живленням обмоток номінальною напругою. При цьому також розширюється діапазон припустимих швидкостей.

Технологія приводів на основі крокових двигунів постійно розвивається. Розвиток спрямований на одержання найбільшого моменту на валу при мінімальних габаритах двигуна, широких швидкісних можливостях, високого ККД і поліпшеної точності. Важливою ланкою цієї технології є застосування мікрокрокового режиму.

На практиці не останнім фактором є й час розробки крокового приводу. Розробка спеціалізованої конструкції для кожного конкретного випадку вимагає значних витрат часу. Із цього погляду вважається більш доцільним застосування універсальних схем керування на основі ШІМ-стабілізації струму, незважаючи на їх більш високу вартість.

Практичний приклад контролера крокового двигуна на основі мікроконтролера сімейства AVR

Незважаючи на те, що в цей час існує велику кількість спеціалізованих мікросхем для керування кроковими двигунами, в окремих випадках можна обійтися й без них. Коли не пред'являється занадто

жорстких вимог, контролер можна реалізувати повністю програмно. При цьому вартість такого контролера виходить дуже низкою.

Пропонований контролер призначений для керування уніполярним кроковим двигуном із середнім струмом кожної обмотки до 2.5 А. Контролер може застосовуватися з розповсюдженими кроковими двигунами типу ДШИ-200-1, -2, -3. Його також можна використати й для керування менш потужними двигунами, наприклад тими, що застосовувалися для позиціонування головок в 5-дюймових дисководах. При цьому схему можна спростити, відмовившись від паралельного включення ключових транзисторів і від ключової стабілізації струму, тому що для малопотужних двигунів досить простого L/R-живлення.

/

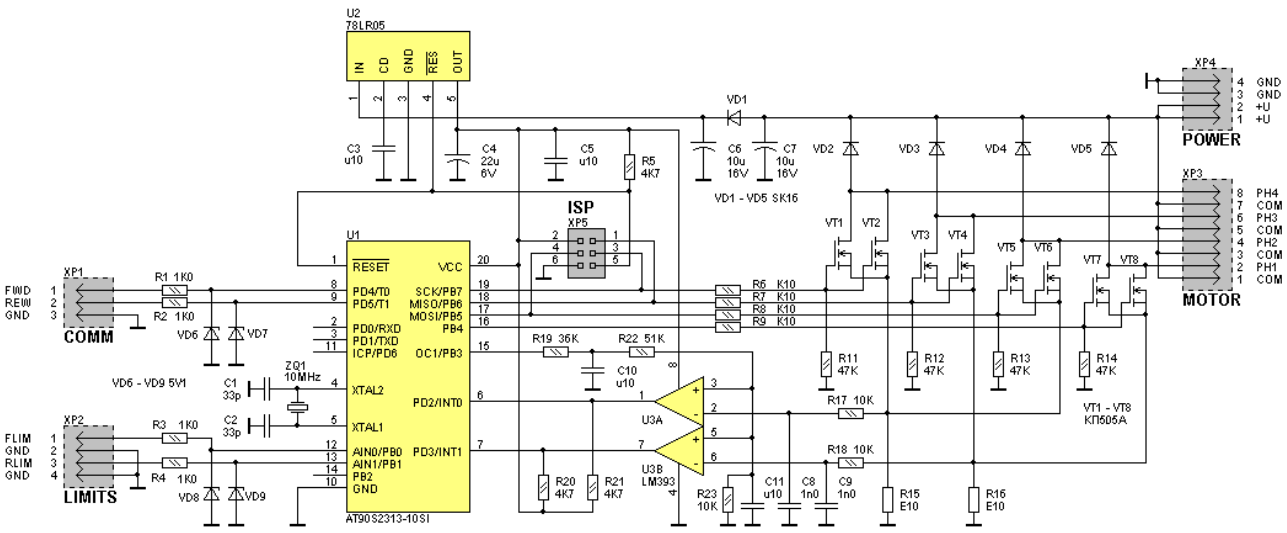


Рис. 1- Принципова схема контролера крокового двигуна

Основою пристрою (рис. 1) є мікроконтролер U1 типу AT90S2313 фірми Atmel. Сигнали керування обмотками двигуна формуються на портах PB4 - PB7 програмно. Для комутації обмоток використовуються по два включених паралельно польових транзистори типу КП505А, всього 8 транзисторів (VT1 - VT8). Ці транзистори мають корпус ТО-92 і можуть комутувати струм до 1.4А, опір каналу становить близько 0.3 Ом. Для того,

щоб транзистори залишалися закритими під час дії сигналу «скидання» мікроконтролера (порти в цей час перебувають у високоімпедансному стані), між затворами й джерелами включені резистори R11 - R14. Для обмеження струму перезарядження ємності затворів встановлені резистори R6 - R9.

Даний контролер не претендує на високі динамічні характеристики, тому цілком влаштовує повільний спад струму фаз, що забезпечується шунтуванням обмоток двигуна діодами VD2 - VD5. Для підключення крокового двигуна є 8-контактний роз’єм XP3, що дозволяє підключити двигун, який має два окремих виводи від кожної обмотки (як, наприклад, ДШИ-200). Для двигунів із внутрішнім з'єднанням обмоток один або два загальних контакти роз’єму залишаться вільними.

Необхідно відзначити, що контролер може бути використаний для керування двигуном з більшим середнім струмом фаз. Для цього тільки необхідно замінити транзистори VT1 - VT8 і діоди VD2 - VD5 на більш потужні. Причому в цьому випадку паралельне включення транзисторів можна не використовувати. Найбільш підходящими є MOS-транзистори, керовані логічним рівнем. Наприклад, такі, як КП723М, КП727У та інші.

Стабілізація струму здійснюється за допомогою ШІМ, що теж реалізована програмно. Для цього використовуються два датчики струму R15 і R16. Сигнали, зняті з датчиків струму, через ФНЧ R17C8 і R18C9 надходять на входи компараторів U3A і U3B. ФНЧ запобігають помилковому спрацьовуванню компараторів внаслідок дії перешкод. На другий вхід кожного компаратора повинна бути подана опорна напруга, яка і визначає максимальний (піковий) струм в обмотках двигуна. Ця напруга формується мікроконтролером за допомогою вбудованого таймера, що працює в режимі 8-бітної ШІМ.

Для фільтрації сигналу ШІМ використовується дволанковий ФНЧ R19C10R22C11. Одночасно резистори R19, R22 і R23 утворюють дільник,

що задає масштаб регулювання струмів фаз. У цьому випадку максимальний піковий струм, що відповідає коду 255, обраний 5.11А, що відповідає напрузі 0.511У на датчиках струму. З огляду на той факт, що постійна складова на виході ШИМ змінюється від 0 до 5 В, необхідний коефіцієнт ділення дорівнює приблизно 9.7. Виходи компараторів підключені до входів переривань мікроконтролера INT0 і INT1.

Для керування роботою двигуна є два логічних входи: FWD (уперед) і REW (назад), підключених до роз'єму XP1. При подачі НИЗЬКОГО логічного рівня на один із цих входів, двигун починає обертатися на заданій мінімальній швидкості і поступово розганяється із заданим постійним прискоренням. Розгін завершується, коли двигун досягає заданої робочої швидкості. Якщо подається команда зміни напрямку обертання, двигун з тим же прискоренням гальмується, потім реверсується й знову розганяється.

Крім командних входів, є два входи для кінцевих вимикачів, підключених до роз'єму XP2. Коли кінцевий вимикач спрацьовує, на відповідному вході присутній НИЗЬКИЙ логічний рівень. При цьому обертання в даному напрямку заборонено. При спрацьовуванні кінцевого вимикача під час обертання двигуна він переходить до гальмування із заданим прискоренням, а потім зупиняється.

Командні входи й входи кінцевих вимикачів захищені від перенапруг ланцюжками R1VD6, R2VD7, R3VD8 і R4VD9, що складаються з резистора й стабілітрона.

Живлення мікроконтролера формується за допомогою мікросхеми стабілізатора 78LR05, що одночасно виконує функції монітора живлення. При зниженні напруги живлення нижче встановленого порога ця мікросхема формує для мікроконтролера сигнал «скидання». Живлення на стабілізатор подається через діод VD1, що разом з конденсатором C6 зменшує пульсації, викликані комутаціями потужного навантаження, який є

кроковий двигун. Живлення на плату подається через 4-контактний роз'єм XP4, контакти якого продубльовані.

Створена демонстраційна версія програми дозволяє здійснювати розгін і гальмування двигуна з постійним прискоренням, а також обертання на постійній швидкості в повнокроковому або напівкроковому режимі. Ця програма містить весь необхідний набір функцій і може бути використана як базова для написання спеціалізованих програм. Тому є сенс розглянути її структуру більш докладно.

Головним завданням програми є формування імпульсних послідовностей для 4-х обмоток двигуна. Оскільки для цих послідовностей тимчасові співвідношення є критичними, формування виконується в оброблювачі переривання таймера 0. Можна сказати, основну роботу програма робить саме в цьому оброблювачі. Блок-схема оброблювача наведена на мал. 2.

Безсумнівно, було б зручніше використати таймер 1, тому що він розрядн-16-розрядний і здатний викликати періодичні переривання по збігу з автоматичним обнулінням. Однак він зайнятий формуванням за допомогою ШІМ опорної напруги для компараторів. Тому доводиться перезавантажувати таймер 0 у перериванні, що вимагає деякого коректування завантажуваної величини і викликає деякий джітер, що, однак, на практиці не заважає.

У якості основної часової бази обраний інтервал 25мкс, що і формується таймером. З такою дискретністю можуть формуватися часові послідовності фаз, такий же період має й ШІМ стабілізації струму у фазах двигуна.

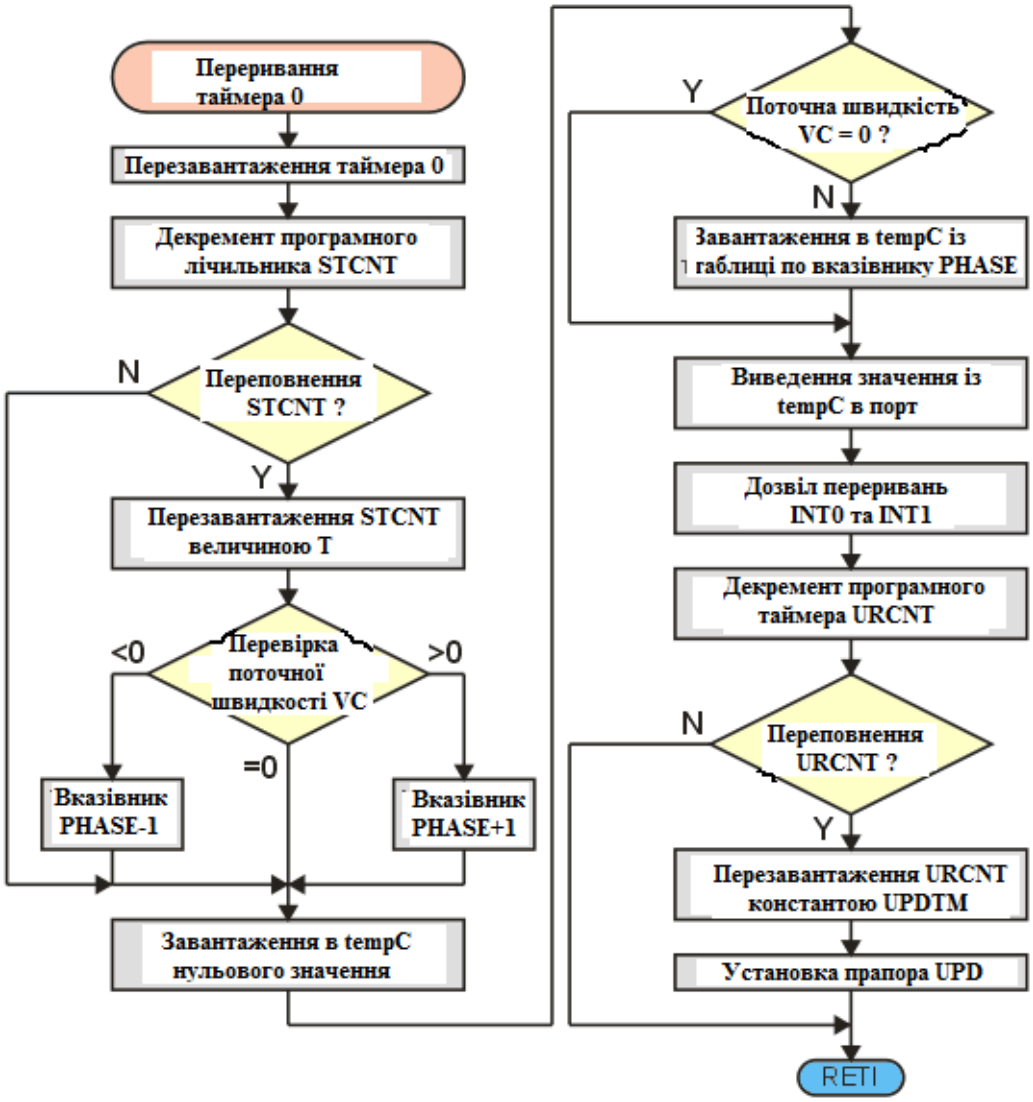


Рис. 2. Блок-схема оброблювача переривання таймера 0

Для формування періоду повторення кроків використовується програмний 16-розрядний таймер STCNT. На відміну від таймера 0, його завантажувальна величина не є константою, тому що саме вона визначає швидкість обертання двигуна. Таким чином, перемикання фаз відбувається тільки при переповненні програмного таймера.

Послідовність чергування фаз задана таблично. У пам'яті програм мікроконтролера є три різні таблиці: для повнокрокового режиму без перекриття фаз, повнокрокового режиму з перекриттям і для напівкрокового режиму. Всі таблиці мають однакову довжину 8 байт.

Потрібна таблиця на початку роботи завантажується в ОЗП, що дозволяє найбільше просто переходити між різними режимами роботи двигуна. Вибірка значень із таблиці відбувається за допомогою покажчика PHASE, тому перемикання напрямку обертання двигуна теж здійснюється дуже просто: для обертання вперед потрібно інкрементувати покажчик, а для обертання назад – декрементувати.

«Головна» змінна в програмі - це 24-бітна знакова змінна VC, що містить значення поточної швидкості. Знак цієї змінної визначає напрямок обертання, а значення - частоту проходження кроків. Нульове значення цієї змінної показує що двигун зупинений. Програма в цьому випадку вимикає струм всіх фаз, хоча в багатьох додатках у цій ситуації потрібно залишити включеними поточні фази й лише трохи зменшити їхній струм, забезпечивши цим утримання положення двигуна. При необхідності таку зміну логіки роботи програми зробити дуже просто.

Таким чином, у випадку переповнення програмного таймера STCNT відбувається аналіз значення змінної VC, у випадку позитивного значення покажчик PHASE інкрементується, а у випадку негативного - декрементується. Потім з таблиці вибирається чергова комбінація фаз, що виводиться в порт. У випадку нульового значення VC покажчик PHASE не змінюється, і в порт виводяться всі нульові значення.

Величина T, яку необхідно завантажувати в таймер STCNT, однозначно зв'язана із значенням змінної VC. Однак перетворення частоти в період займає досить багато часу, тому ці обчислення виконуються в основній програмі, і не на кожному кроці, а набагато рідше.

Взагалі, ці обчислення потрібно періодично робити тільки під час розгону або гальмування. В інших випадках швидкість, і, відповідно, період повторення кроків, не міняються.

Контрольні питання

1. Які фактори визначають максимальний момент і потужність крокового двигуна?
2. Яким чином задається чергування фаз у контролері?
3. Як виконується регулювання струму у схемі контролера?
4. Поясніть логіку роботи кінцевих вимикачів у схемі контролера.
5. Як виконується фільтрація сигналів ШІМ у схемі контролера?

Лекція 18

ПРОГРАМНЕ КЕРУВАННЯ СТРУМОМ ДВИГУНА

Для здійснення ШІМ-стабілізації струму фази повинні періодично вмикатися, а потім, при досягненні струмом заданого рівня, вимикатися. Періодичне вмикання виконується в перериванні таймера 0, для чого навіть у випадку відсутності переповнення програмного таймера STCNT у порт виводиться поточна комбінація фаз. Відбувається це з періодом 25мкс (що відповідає частоті ШІМ 40 кГц). Вимиканням фаз управляють компаратори, виходи яких підключені до входів переривання INT0 і INT1. Переривання дозволяються після того, як фази вмикаються, і забороняються відразу після перемикання компараторів. Це виключає їхню повторну обробку. В оброблювачах переривань відбувається тільки відключення відповідних фаз (рис. 1).

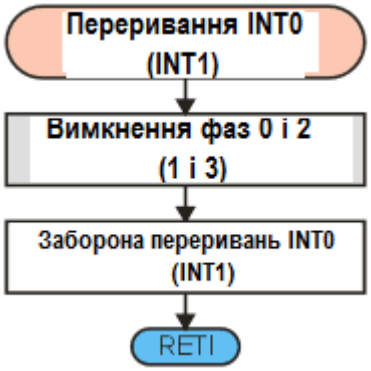


Рис. 1- Блок-схема оброблювача переривань INT0 і INT1

Процеси, що відбуваються при ШІМ-стабілізації струмів, показані на рис. 2. Особливо слід зазначити, що вихідний сигнал від датчика струму має переривчастий характер навіть у тому випадку, якщо струм обмотки не переривається. Це зв'язано тим, що під час спаду струму його шлях не проходить через датчик струму (а проходить через діод).

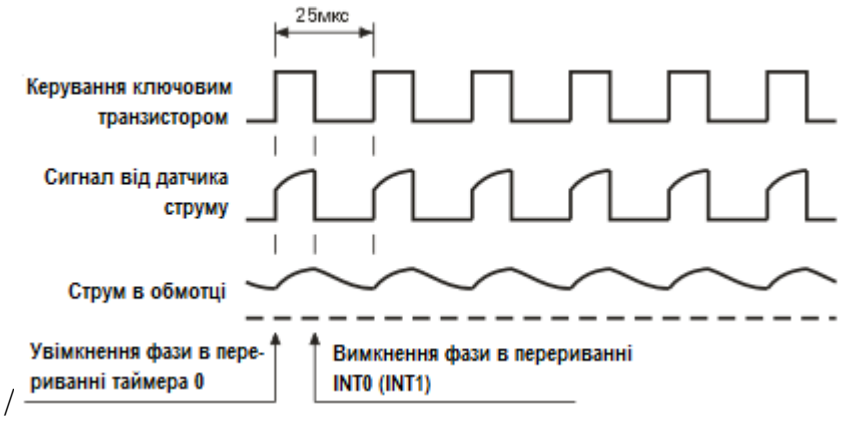


Рис. 2- Процес ШІМ-стабілізації струму

Потрібно відмітити, що аналогова частина системи ШІМ-стабілізації струму фаз двигуна є досить «примхливою». Справа в тому, що сигнал, що знімається з датчика струму, містить велику кількість перешкод. Перешкоди виникають в основному в моменти комутації обмоток двигуна, причому як «своєї», так і «чужої» фази.

Для правильної роботи схеми потрібне коректне розведення друкованої плати, особливо це стосується земляних провідників. Можливо, доведеться підібрати номінали ФНЧ на вході компаратора або навіть ввести в компаратор невеликий гістерезис.

Як ми вже відзначали вище, при керуванні малопотужними двигунами від ШІМ-стабілізації струму можна зовсім відмовитися, застосувавши звичайну L/R-схему живлення обмоток. Для вимкнення ШІМ-стабілізації досить просто не підключати входи INT0 і INT1 мікроконтролера, природно, при цьому можна взагалі не встановлювати компаратор і датчики струму.

У даній програмі періодичність обчислення нових значень швидкості й періоду обрана рівною 15.625мс. Таке значення обране не випадково. Цей інтервал становить 1/64 с, а головне, він містить ціле число періодів переповнення таймера 0 (25мкс). Зручно, якщо значення швидкості й прискорення задаються в природних одиницях, тобто в кроках у секунду й у кроках, ділених на секунду у квадраті. Для того щоб мати можливість у

цілочисельній арифметиці обчислювати миттєву швидкість 64 рази в секунду, потрібно перейти до внутрішнього подання швидкості, збільшеного в 64 рази. Множення й ділення на 64 зводиться до звичайних зсувів і тому вимагає дуже мало часу. Задану періодичність обчислень забезпечує ще один програмний таймер URCNT, який декрементується в перериванні таймера 0 (один раз за 25 мкс). Цей таймер завжди завантажується постійною величиною, що забезпечує незмінний період його переповнень, рівний 15.625мс. При переповненні цього таймера встановлюється бітовий прапор UPD, що сигналізує основній програмі, що необхідно обновити значення швидкості й періоду.

Основна програма (рис. 3) виконує обчислення миттєвих значень швидкості й періоду проходження кроків, забезпечуючи необхідну часову діаграму розгону. У цьому випадку розгін і гальмування здійснюються з постійним прискоренням, тому швидкість міняється лінійно. Період при цьому міняється за гіперболічним законом, і його обчислення - основна робота програми.

Оновлення значень швидкості й періоду проходження кроків основна програма робить періодично, періодичність задається прапором UPD. Оновлення програма робить на основі порівняння значень двох змінних: миттєвої швидкості VC і необхідної швидкості VR.

Значення необхідної швидкості також визначається в основній програмі. Це робиться на основі аналізу керуючих сигналів і сигналів з кінцевих вимикачів. Залежно від цих сигналів, основна програма завантажує змінну VR значенням необхідної швидкості. У даній програмі це V для руху вперед, -V для руху назад і 0 для зупинки. У загальному випадку, набір швидкостей (а також прискорень і струмів фаз) може бути як завгодно великим, залежно від вимог.

Якщо швидкості VC і VR рівні, виходить, крановий двигун працює в стаціонарному режимі й оновлення не потрібно. Якщо ж швидкості не рівні, то значення VC із заданим прискоренням наближається до VR, тобто двигун прискорюється (або сповільнюється) до досягнення номінальної швидкості. У випадку, коли навіть знаки VR і VC відрізняються, двигун сповільнюється, реверсується й потім досягає необхідної швидкості. Відбувається це автоматично завдяки структурі програми.

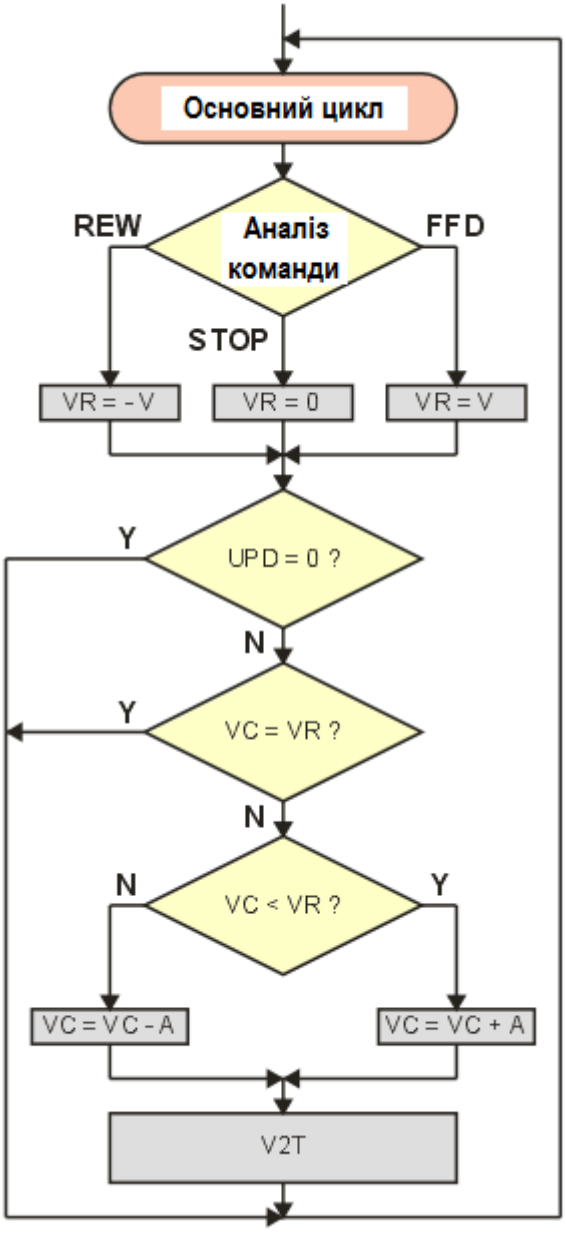


Рис. 3- Блок-схема основного циклу програми

Якщо при черговій перевірці виявляється, що швидкості VR і VC не рівні, то до значення VC додається (або віднімається) значення прискорення A. Якщо в результаті цієї операції відбувається перевищення необхідної швидкості, то отримане значення коректується шляхом заміни на точне значення необхідної швидкості.

Потім відбувається обчислення періоду T (рис. 4).

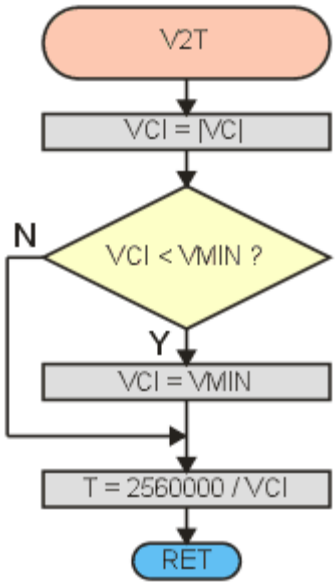


Рис. 4 - Блок-схема підпрограми обчислення періоду

Спочатку обчислюється модуль поточної швидкості. Потім відбувається обмеження мінімальної швидкості. Це обмеження необхідно з двох причин.

По-перше, нескінченно малій швидкості відповідає нескінченно великий період, що викличе помилку в обчисленнях. По-друге, крокові двигуни мають досить протяжну по швидкості зону старту, тому немає необхідності стартувати на дуже маленькій швидкості, тим більше що робота на малих швидкостях викликає підвищений шум і вібрацію.

Значення мінімальної швидкості VMIN необхідно вибирати виходячи з конкретного завдання й типу двигуна. Після обмеження мінімальної швидкості виконується обчислення періоду по формулі T = 2560000/|VC|.

На перший погляд формула не очевидна, але якщо врахувати, що період необхідно одержати в 25 мкс-інтервалах, а внутрішнє представлення VC - це помножене на 64 її значення, те все стає на свої місця. При обчисленні T потрібна операція беззнакового ділення формату 24/24, що мікроконтролер AVR на тактовій частоті 10 Мгц робить приблизно за 70 мкс.

З огляду на те, що обчислення періоду відбуваються не частіше, ніж один раз в 15.625мс, завантаження процесора виходить дуже низким. Основне завантаження робить переривання таймера 0, та й воно в основному виконується по короткій вітці (без переповнення STCNT) тривалістю приблизно 3мкс, що відповідає 12%-й завантаженню процесора.

Це означає, мікроконтролер має значні резерви обчислювальних ресурсів.

Контрольні питання

1. Поясніть особливості виконання аналогової частини системи ШІМ.
2. Поясніть алгоритм роботи основного циклу програми.
3. Поясніть алгоритм роботи підпрограми обчислення періоду.
4. Поясніть причини обмеження мінімальної швидкості.
5. Як у програмі виконується множення і ділення на 64?

Список літератури

1. Алексеев К.Б., Палагута К.А. Микроконтролерное управлене электроприводом: Учебное пособие.- М.: МГИУ, 2008. – 298 с.
2. Брусникин Д.Э., Зорохович А.Е., Хвостов В.С. Электрические машины: В 2-х ч. М.: Высшая школа. 1987.
3. Королев Н. AVR-микрокортролеры второго поколения: новые аппаратные возможности // Электронные компоненты. 2003, № 4.
4. Козаченко В.Ф. Основные тенденции развития встроенных систем управления двигателями и требования к микроконтроллерам // Chip News. 1999. № 1.
5. Микропроцессорные системы: Учебное пособие для вузов / Под редакцией Д.В.Пузанкова. – М.: Политехника, 2002.
6. Евстифеев А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. – М. Издательский дом «Додэка-XXI», 2007.
7. Программирование на языке С для AVR и PIC микроконтроллеров./ Сост. Ю.А. Шпак – К.: - «МК-Пресс», 2006. – 400 с.

ЗМІСТ