trans & MOTAUTO'05+

BLDC Software Control

ПРОГРАМИРОВАННЫЙ КОНТРОПЬ BLDC

Teaching Assi. Eng. Aghion C.[1], Lect. Eng. Ursaru O.[2] , Assoc. Prof. Dr. Eng.Vornicu L.[3],
Faculty of Electronics and Telecommunications Technical University of Iasi -Romania
E-mail: aghion@etc.tuiasi.ro; E-mail: ovidiu@etc.tuiasi.ro

Abstract: *In this paper, we suggest a PWM command for trapezoidal BLDC engine, using Hall sensors, in current closed loop, with PID reaction. The software command is implemented on a C8051F120 microcontroller made by Silicon Laboratories, which produces a PWM command at 24KHz carrier frequency.*
***KEYWORDS**: MICROCONTROLLER, SOFTWARE, PID, COMMAND, POWER*

1. Introduction

BLDC engines (Brushless DC) are used in high speed applications, canning and bottling, machine tools, material handling, robotics, compressors, fans, treadmills, etc, with some advantages (low cost, precise speed control, moderate PWM losses) and disadvantages (high torque ripple, medium performance, high radial forces on motor). The application was successfully tested on the BLDC engine, type BM10, equipped with Hall sensors, made by PennEngineering Motion Technologies Company.

2. Results and discussion

Fig. 1 shows the simplified circuit used for this software.

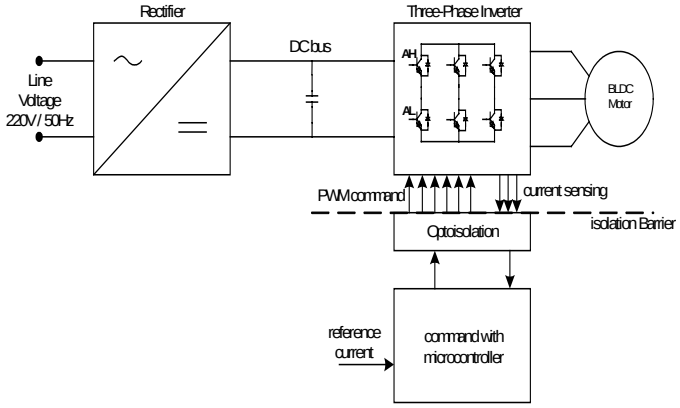

Fig. 1 Simplified circuit diagram

The most important block for this application is the C8051F120 microcontroller, used to work at 96MHz, with the following inputs:

- 3 buttons (Start, Stop, Reverse Engine);
- 3 sense inputs (i_A, i_B, i_C);
- 3 sensors Hall inputs (H_A, H_B, H_C);

and outputs:

- 3 PWM commands (for the upper side of the three phase power inverter);
- 3 normal command outputs (for lower side of the three phase power inverter);

Fig. 2 presents the following waveforms:

- Hall sensors outputs (H_A, H_B, H_C)
- output torque
- sense outputs currents

2.1 Software description

First of all we must initialize the microcontroller: Oscillator, Timer T0, T1, Watchdog, PCA, ADC0, Ports and

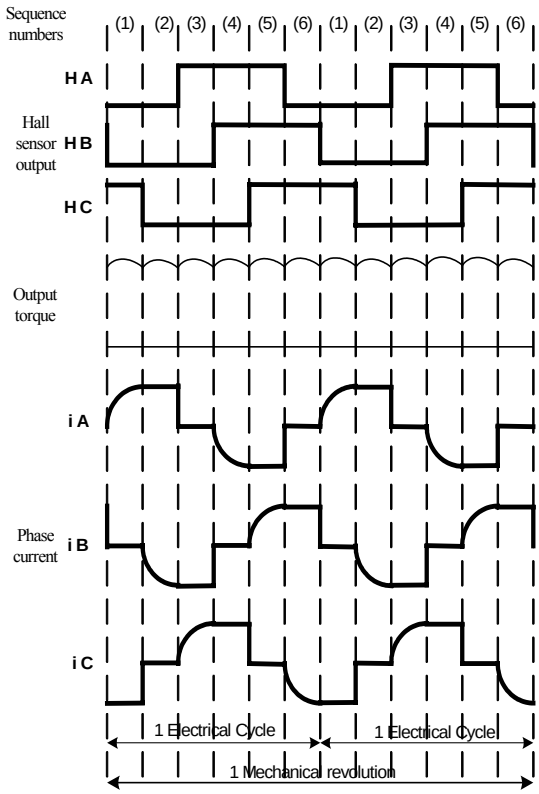

Fig. 2 Waveforms of the Hall sensors, output torque, phase current

Interrupts, which are realized in Initialize_Device routine. After that, we read the offsets of the sense currents (without inverter bridge enabled) - (i_A, i_B, i_C) and the Hall sensor output (in order to know the position of the rotor).
(see Fig. 3)

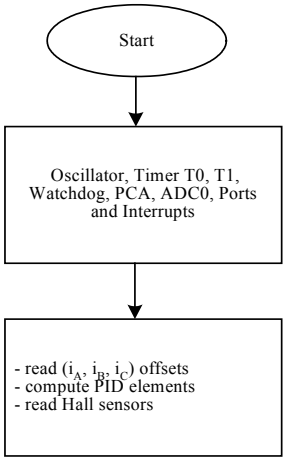

Fig. 3

In Main loop we just read the buttons and the reference current (used for the implementation of the PID control algorithm). (see Fig. 4)

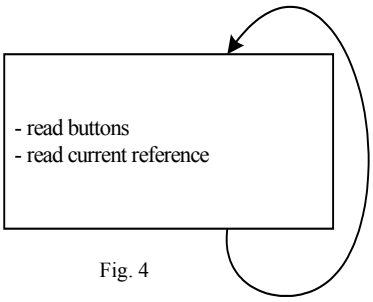

Fig. 4

At every 50 microseconds the Timer 3 generates an interruption that leads in 32 microseconds to the execution of the interruption routine, which is the most important routine in this program. At the beginning of this routine, tests are made in order to determine the conduction transistors in the inverter, so that a single current can be read, that is the current with the positive amplitude. From this value we substract the current offset that we read at the initialization. At the beginning of this routine the START button is tested; if it wasn't pressed the inverter bridge is deactivated. Then the control current closed loop is implemented (PID), which determines at the end of this routine the calculation of the duty factor for the three PWM outputs corresponding to the upper transistors from the inverter bridge. The calculation of the duty factor starts from the nominal voltage of the engine and takes into account the maximum load current of the PWM block. (because this block is set to work with self load, at 8 bits). After the calculation of the duty factor, the position of the rotor is read from the Hall sensors, then the rotation bit flag is tested and finally the "commutation" routine is called, which has the role of commanding the transistors from the three phase power inverter. We must mention that for every position of the Hall sensors corresponds a commutation position of the three phase power inverter, which was memorized in a commutation vector. At the end of the Interruption service routine we can test the value of the sense current, and if this value is higher then the nominal current of the three phase power inverter, the bridge inverter will be deactivated.

Fig. 3 presents the PID command technique used in this software, which, first of all, was simulated with the Matlab program.

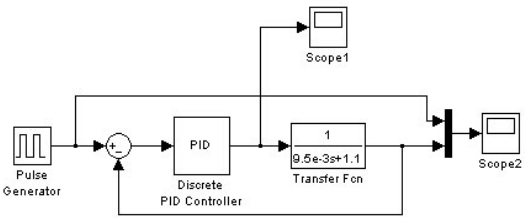

Fig. 3 The simulation of the PID command

Fig. 4a shows the output of the PID block response and 4b shows the output of the entire PID command, using the function transfer of the Brushless DC engine:

$$\frac{1}{9.5e-3 \quad + \quad 1.1}.$$

The PID block was implemented using the following equations:

Fig. 4a

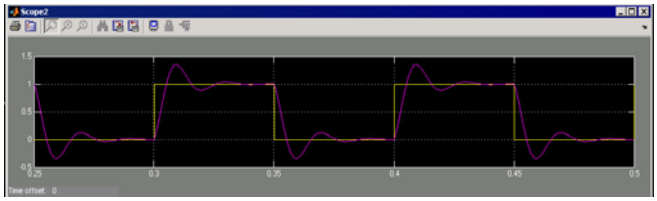

Fig. 4b

$$G_R(s) = k_R\left(1+\frac{1}{T_i s}+sT_d\right)=\frac{U(s)}{g(s)}, \tag{1}$$

where:
k_R = the proportional factor
T_d = the derivation time
T_i = the integration time
The ideal equation of the PID algorithm is:

$$u(t)=k_R\left[g(t)+\frac{1}{T_i s}\int_0^t g(\tau)d\tau+T_d\frac{dg(t)}{dt}\right]. \tag{2}$$

Using the rectangle form, we obtain:

$$u(k)=k_R\left\{g(k)+\frac{T}{T_i}\sum_{j=1}^{k}g(i)+\frac{T_d}{T}[g(k)-g(k-1)]\right\}. \tag{3}$$

To obtain a recursive algorithm, we calculate u(k-1):

$$u(k-1)=k_R\left\{g(k-1)+\frac{T}{T_i}\sum_{j=1}^{k-1}g(j)+\frac{T_d}{T}[g(k-1)+g(k-2)]\right\} \tag{4}$$

Substracting the last two equations, we obtain:

$$\Delta u(k)=u(k)+u(k-1)=$$
$$k_R\left\{g(k)-g(k-1)+\frac{T}{T_i}g(k)+\frac{T_d}{T}[g(k)-2g(k-1)+g(k-2)]\right\} \tag{5}$$

or:

$$u(k)=u(k-1)+q_0\varepsilon(k)+q_1\varepsilon(k-1)+q_2\varepsilon(k-2), \tag{6}$$

with the following parameters:

$$q_0=k_R\left(1+\frac{T}{T_i}+\frac{T_d}{T}\right),\quad q_1=-k_R\left(1+2\frac{T_d}{T}\right),\quad q_2=k_R\frac{T_d}{T}. \tag{7}$$

We can test the stability of the algorithm, with the following equations:

$$q_0>0,\quad q_1<-q_0,\quad -(q_0+q_1)<q_2<q_0. \tag{8}$$

2.2 Software implementation

```
void Timer3_ISR (void) interrupt 14
{
   unsigned char h;
   SFRPAGE = CONFIG_PAGE;
            // Measure the stator current on the commanded phase
   if (LINP1 == 0)
   {
      crt = current_measure(0);
      crt_zeroval = crt_zeroval_u;
   }
            else
   if (LINP2 == 0)
   {
      crt = current_measure(1);
      crt_zeroval = crt_zeroval_v;
   }
            else
   if (LINP3 == 0)
   {
      crt = current_measure(2);
      crt_zeroval = crt_zeroval_w;
   }

            if(crt < crt_zeroval)
                     crt = 0;
            else
               crt = crt - crt_zeroval;

   if(crt_ref>0)
                     {
            if(start==0)
                     // Stop the motor
                     {
                     crt_ref = 0;
                     if(pid_out == 0)
                              {
                     SFRPAGE = TMR3_PAGE;
                     TR3 = 0;
                     TF3 = 0;
                     disable_command();
                     pid_reset();
                     main_flag = 0;
                     crt_ref_temp = 0;

                     if(brake)
                                       {
                              delay_ms(3000);
                              SFRPAGE = CONFIG_PAGE;
                              LEDGI = 1;
                              SFRPAGE = PCA0_PAGE;
                              PCA0CPH3 = block_dc;
                              PCA0CPH2 = block_dc;
                              PCA0CPH1 = block_dc;
                     h = hallPosition();       // h equals hall position
            commutate(h);       // commutate motor, enables PWM
                                 SFRPAGE = CONFIG_PAGE;
                                          P_EN_C = 0;
                                                   return;
                                       }
                              }
                     }
                     else
                     {
                        SFRPAGE = CONFIG_PAGE;
                              LEDGI = 0;

                              if(P_EN_C == 1)
                              {
                                       pid_reset();
                                                P_EN_C = 0;
                              }
                     }

                     // change rotation sense
                     if (reverse)
                     {
                              crt_ref = 0;
                              if(pid_out == 0)
                              {
                              SFRPAGE = TMR3_PAGE;
                              TR3 = 0;
                              disable_command();
                              pid_reset();
                              delay_ms(3000);
                              reverse = 0;
                              rot_sense = ~rot_sense;
                              SFRPAGE = PCA0_PAGE;
                              P_EN_C = 0;
                              SFRPAGE = TMR3_PAGE;
                              TR3 = 1;
                              }
                     }

            // don't allow current setpoint to change suddenly
                     if(crt_ref > crt_ref_old)
                     {
                              if (pid_cnt == MAX_PID_CNT)
                                       crt_ref = crt_ref_old + 1;
                              else
                                       crt_ref = crt_ref_old;
                     }
                     else
                     if(crt_ref_old > crt_ref)
                     {
                              if (pid_cnt == MAX_PID_CNT)
                                       crt_ref = crt_ref_old - 1;
                              else
                                       crt_ref = crt_ref_old;
                     }

                     pid_cnt++;
                     if (pid_cnt > max_pid_cnt)
                              pid_cnt = 0;

crt_ref_old = crt_ref;
crt = (int)(((long)crt*5553L) >> 8); // Convert measured current to
mA

            eps_1 = eps_0;       // Store previous error value
            eps_0 = crt_ref - crt;// Compute current error value
            integral += eps_0;    // Compute integral term
            if(integral > max_integral)// Avoid integral term overflow
            integral = max_integral;
            else
                     if(integral < -max_integral)
                     integral = -max_integral;
            derivative = eps_0 - eps_1;// Compute derivative term
                     // Compute PID output
                     pid_out =  (((long)eps_0*kr)) >> 10;
                     pid_out += (((long)integral) * ki) >> 10;
                     pid_out += (((long)derivative*kd)) >>10;

         // Limit PID output
         if (pid_out > max_dc)
            pid_out = max_dc;
         if (pid_out < 0)
            pid_out = 0;
```

```
        pid_out_old = pid_out;
    }
    else
    {
                        disable_command();
                        pid_reset();
    }


    SFRPAGE = PCA0_PAGE;
    PCA0CPH3 = pid_out;
    PCA0CPH2 = pid_out;
    PCA0CPH1 = pid_out;

    h = hallPosition();        // h equals hall position
    commutate(h);              // commutate motor, enables PWM

            main_flag = 0;
            SFRPAGE = TMR3_PAGE;
            TF3  = 0;
}
```

3. Conclusions

This software is very useful for the development of easy and safe applications where a current closed loop is required. After the calculation the parameters kr, ki and kd (using formulas) and the simulation in Matlab program, it is easy to introduce these values in the software as global constants (if it is not necessary to change these values when the rotor is still in move) and to see the feedback on the prototype platform. In our case, we give an external current value to the microcontroller, which tries to control the sense current (obtained from the load) at the same value imposed by us, according to the parameters kr, ki and kd.

4. References

[1] Silicon Laboratories, C8051F120 – Small Form Factor.
[2] Phillips Semiconductors 80C51-Based 8-Bit Microcontrollers
[3] Texas Instruments,Data Book, 1992.
[4] Siemens, Data Book, ICs for Communications, 1992.
[5] International Rectifier – Motor Drives Applications.
[6] Microchip – Brushless DC Motor Control Made Easy – AN857
[7] Microchip – Brushless DC Motor Control Using PIC18FXX31 MCUs - AN899
[8] Petrut Duma, "Microcontrolerul INTEL8051 Aplicatii", ETP Tehnopress, Iasi-2003.
[9] N. Mohan, T.M. Undeland, W. P. Robbinson, “Power Electronics: Converters Applications and Design,” Willey & Sons, New York, 1989.
[10] C. Lazar, “Ingineria Reglarii Automate – Partea a II-a”, on-line courses at: http://www.ac.tuiasi.ro/ro/library/ira/ira.html