AC Servo Motor Controller Based on TMC4671 Break-Out Board

B. A. V. Perera - LE Robotics (Pvt.) Ltd., J.A.K.S. Jayasinghe – University of Moratuwa

Abstract:- **Modern servo motor controllers capable of precision position, velocity and torque control have massively boosted up many industries such as robotics, food production, military, etc. during the past decades [1]. High cost of servo motor control systems due to the complexity and the high-end technical expertise required to design, operate and maintain these systems can be a major problem in certain applications. Therefore, cost effective and easy-to-operate systems are needed. This paper presents the use of TMC4671-BOB module based on the Field Oriented Control (FOC) algorithm from TRINAMIC Motion Control GmbH & Co. KG for a cost effective and easy-to-operate servo motor controller. Configuring the module to operate a servo motor with an incremental encoder in Velocity and Position control modes is presented.**

Keywords:- **Field Oriented Control (FOC), Servo Motor, Velocity Control, Position Control**

I. INTRODUCTION

The concept of Field Oriented Control (FOC) or vector control was developed independently by K. Hasse [2], and by Felix Blaschke [3] in late 1960's and early 1970's respectively [4]. The FOC algorithm is a current regulation scheme for driving AC synchronous motors using two orthogonal components. One orthogonal component (which is usually referred to as I_q) generates the torque while the other orthogonal component (which is usually referred to as I_d) generates the back emf. By using the FOC algorithm, AC synchronous motors are controlled by regulating the I_q component (while keeping the $I_d = 0$ usually) to obtain the required torque like a DC brushed motor. In order to obtain the I_d and I_q, the rotor position and phase currents are required. The rotor position is usually obtained by an encoder while the currents of two phases are measured and transformed into a synchronously rotating frame using well known Clarke-Park transformation. The Clarke-transformation transforms the stator current vector ($\bar{I}_s$) created by the 3 phase currents ($\bar{I}_a$, $\bar{I}_b$, $\bar{I}_c$) into a bi-phased currents $\bar{I}_\alpha$, $\bar{I}_\beta$ shown in Fig. 1.

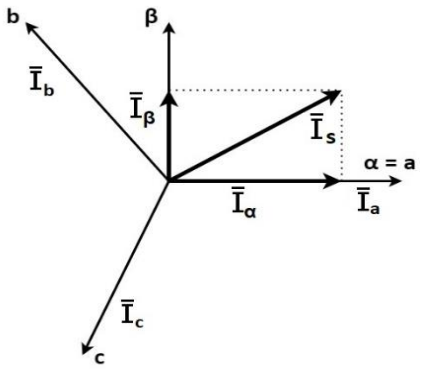

Fig. 1 - abc Frame to αβ Frame Transformation [5]

The Park-transformation transforms the output of the Clarke transformation into 2-phase synchronous rotating dq frame which rotates at the electrical speed of the rotor such that the d axis is aligned with the rotor flux ($\bar{\psi}_R$) as shown in Fig. 2. (Refer [5] for the respective transformation equations.)

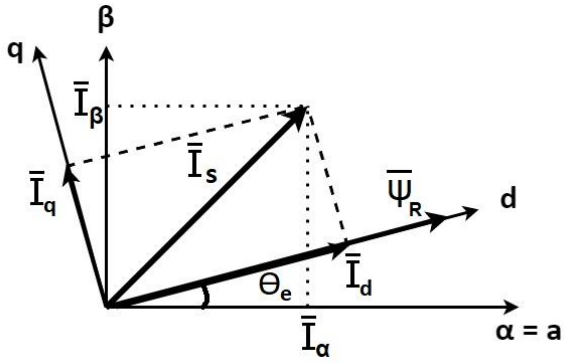

Fig. 2 – αβ Frame to dq Frame Transformation [5]

Fig. 3 shows the FOC control structure for the position control of an AC servo motor. Clarke and Park transformation are applied to the measured I_a, I_b phase currents as described above. Then I_d, I_q are compared with reference values 0 and target torque T_T to generate error signals for two independent PI current controllers. The outputs of these two PI current controllers are passed through an inverse Clarke-Park transformation stage to obtain stator voltages U_u, U_v and U_w. Then, these three stator voltages are taken as reference waveforms for synthesizing high voltage three phase signals from a high voltage DC source using the Space Vector Modulation (SVM) technique [5]. The outer most position loop drives the velocity target V_T and torque target T_T to reach required the position target P_T using two more PI controllers [5].

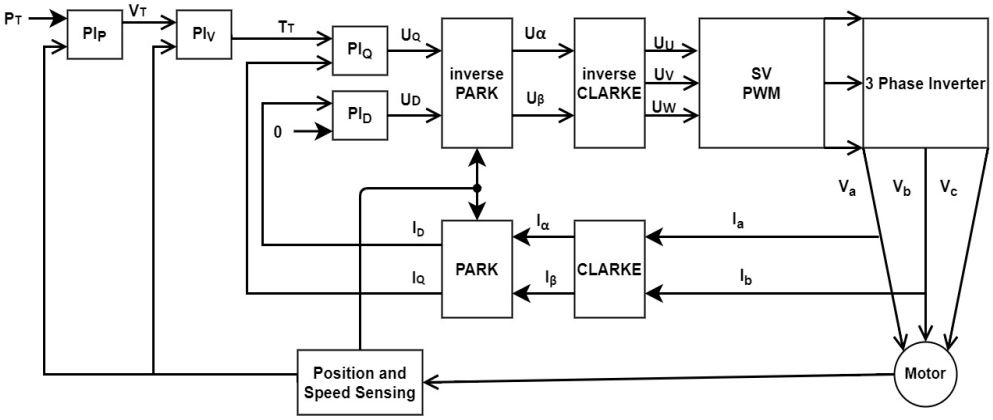

Fig. 3 - PMSM Position Control FOC Structure

The FOC algorithm implementation became more popularized commercially with the development of DSP chips such as TMS32010 [6]. With the first generation of DSP chips, external peripherals such as ADCs were connected and the algorithm was coded in assembly language to achieve the required computational speeds. Even with the modern DSP's such as dsPIC33F [7] with on-chip ADCs, developing codes for the FOC algorithm is a tedious task. This issue was addressed and the control algorithm was implemented in Application Specific ICs (ASICs) like TMC4671-LA. With these ASICs, servo motors can be controlled very effectively by accessing a set of registers for configuring the FOC algorithm. Use of TMC4671-BOB containing the TMC4671-LA relieved us form designing a complex PCB for the fine-pitched package of the TMC4671-LA.

The TMC4671-BOB (see Fig. 4) allows user to connect incremental encoder or hall sensor to obtain the rotor position and provides two ADC channels to measure phase currents and one ADC channel to observe the bus voltage of the motor driver. Since the TMC4671-LA registers needs to be programmed when powering up, the TMC4671-BOB module allows to communicate with the TMC4671-LA chip using SPI or USART interface. This module is further equipped with a Real Time Monitoring Interface for debugging and tuning the PI controllers using Trinamic RTMI adapter and TMCL IDE software [8].

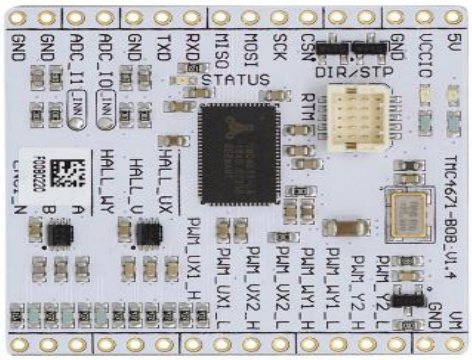

Fig. 4 - TMC4671-BOB [8]

II. COMMUNICATION WITH THE TMC4671-BOB AND REGISTER LIST

A protocol with a 5-byte data packet is used with the SPI and USART interfaces for sending and receiving data to the TMC4671-LA chip. It always returns a 5-byte packet for each receiving 5-byte packet. The most significant byte of the data packet represents the register address and the following 4 bytes contains the data which should be written to the chip in the write mode or the data read from a specified register in the read mode. If the data is written to the module, the most significant bit in the most significant byte (address byte) should be raised to one (write mode), else it should be zero (read mode). Hence whenever the data is written to the chip, 0x80 is added to the register address. Before programing the TMC4671-LA chip, it is essential to check the communication link and the working condition of the TMC4671-LA chip. This is done by sending **0x00000000** to the TMC4671-LA chip which returns **0x0034363731** on success.

Table 1 describes the register list used in TMC4671-LA for setting up of position and velocity control modes. Use of these registers in proper sequence will be described in Sections III to VIII.

Register	Description	Access	Usage
0x04	Used to configure internal ADCs	RW	Init
0x08	Used to calibrate ADC channel1	RW	Init
0x09	Used to calibrate ADC channel0	RW	Init
0x0A	Used to assign ADC channel to PWM output channel	RW	Init
0x17	Used to set the polarity of PWM output signal	RW	Init
0x18	Used to configure PWM output frequency	RW	Init
0x19	Used to set dead-time	RW	Init
0x1A	Used to enable PWM	RW	Init
0x1B	Used to set motor type and number of pole pairs	RW	Init
0x1C	Used to set electrical angle when the encoder is not connected to TMC4671 during initialization	RW	Test
0x1F	Used to change direction of open loop angle	RW	Init
0x20	Used to change open loop acceleration	RW	Init
0x21	Used to set target velocity for open loop angle generator	RW	Init
0x23	Used to displays actual output of open loop angle generator	RW	Monitor/Test
0x24	Used to set voltages for open loop current control mode	RW	Init /Test
0x25	Used to configure decoder input signals	RW	Init
0x26	Used to set PPR value of the encoder	RW	Init
0x29	Used to set offsets for electrical and mechanical angle calculated from encoder	RW	Init
0x2A	Used to read actual angles from ABN encoder	R	Monitor
0x50	Used to select the source for velocity measurement	RW	Init
0x51	Used to select the source for position measurement	RW	Init
0x52	Used to select an angle signal for FOC transformation	RW	Init
0x54	Used to set control parameters for flux controller	RW	Init
0x56	Used to set control parameters for torque controller	RW	Init
0x58	Used to set control parameters for velocity controller	RW	Init
0x5A	Used to set control parameters for position controller	RW	Init
0x5E	Used to set target current limit for flux and torque controllers	RW	Init
0x60	Used to set an absolute velocity limit for velocity controller input	RW	Init
0x63	Used to set motor motion mode	RW	Init
0x64	Used to set target values for torque and flux controllers	RW	Control
0x66	Used to set target velocity for velocity controller	RW	Control
0x68	Used to set target position for position controller	RW	Control
0x6B	Used to read actual position derived from position signal	RW	Monitor/ Init
0x78	Used to set step width of an actual input step signal on STEP/DIR interface	RW	Init

Table 1 - Register List Used in Programming TMC4671-LA [4]

III. MOTOR PARAMETERS AND PWM SETTINGS

The datagrams shown in Table 2 will initialize the motor type and enable the PWM with dead-time of 25ns.The datagrams listed in Table 2 is common for Position and Velocity control mode.

Register	Datagram	Description
0x1B	0x9B00030004	Motor type = 3 (three phase BLDC motor) and number of pole pairs = 4
0x17	0x9700000000	Set PWM polarity
0x18	0x9800000F9F	PWM frequency = 25kHz
0x19	0x9900001919	Dead-time = 250 ns

Table 2 - Motor Parameter and PWM Settings

IV. ADC SETTINGS

The datagrams shown in Table 3 will set the sampling frequency and assign ADC channels for current measurement. Current offsets should be measured by using TMCL IDE and write to the 0x08 and 0x09 registers. The datagrams listed in Table 3 is common for both operating modes indicated above.

Register	Datagram	Description
0x04	0x8400100010	Delta-Sigma Modulator sampling frequency = 100MHz
0x0A	0x8A24000100	Select physical ADC channels and assign channels
0x08	0x8801007FFF	ADC1 offset correction
0x09	0x8901007FFF	ADC0 offset correction

Table 3 - ADC Settings

V. INCREMENTAL ENCODER INITIALIZATION SETTINGS

It is essential to bring the rotor to an initial known starting position and initialize the encoder position correctly for the incremental encoders, otherwise calculations of angles, velocity and position will be faulty. By executing the boot sequence listed in Table 4, the rotor can be aligned with the magnetic field and initialize the encoder using the open loop mode. During this initialization process, it is important to set the exact encoder resolution as well as the encoder rotation direction. In this table, it is assumed that, a 500 ppr quadrature encoder is connected. Hence, the number of pulses for a mechanical revolution becomes 500x4, which is 2000 (7D0 in HEX). The encoder resolution can be set by accessing the 0x26 register. Setting the encoder direction should be done according to the encoder placement of the motor. Encoder direction setting can be done by accessing 0x25 register. The datagrams listed in Table 4 is common for both operating modes as indicated above. All the datagrams up to the last two rows in Table 4 are sent as a block and the rotor will be aligned after sending the command sequence. The mechanical angle offset should be corrected before activating the operating mode. This is done by last two commands of Table 4. Sending the 0x2A00000000 command reads the electrical (1st & 2nd bytes) and mechanical angle (3rd & 4th bytes) of the rotor. Then write the 2's complement of the mechanical angle to the 3rd and 4th bytes of the last command of Table 4 to correct the rotor offset. Execute the 0x2A00000000 command again to check whether the mechanical angle is 0. If the mechanical angle is 0, then the desired operating mode can be activated.

Register	Datagram	Description
0x1F	0x9F00000000	Rotation direction
0x20	0xA00000003C	Rate of change the velocity = 60 rpm/min
0x21	0xA100000000	Open loop velocity target = 0
0x23	0xA300000000	Open loop angle = 0
0x52	0xD200000000	Default setting
0x63	0xE300000008	Open loop mode selection
0x26	0xA6000007D0	Encoder pulse per mechanical revolution = 2000 ppr
0x29	0xA900000000	Electrical and mechanical angle offset = 0
0x52	0xD200000001	Select phi_e_ext
0x24	0xA400000FA0	Initialization voltage on UD_EXT = 4000
0x1C	0x9C00000000	Set the zero angle
0x25	0xA500001000	Encoder direction clockwise
0x52	0xD200000000	Set PHI_E to 0 so the rotor aligns itself with it once PWM enabled
0x1A	0x9A00000107	Enable PWM
0x2A	0x2A00000000	Read angles
0x29	0xA90000xxxx	2's complement of current mechanical angle = xxxx

Table 4 - Incremental Encoder Initialization Settings

VI. VELOCITY MODE SETTINGS

The boot sequence listed in Table 5 will control the motor in velocity control mode for target velocity of 100 rpm. In velocity control mode, it is essential to have a source to calculate the velocity. The mechanical encoder angle was selected as the angle source by accessing the 0x50 register. The target velocity should be set before the velocity mode selection to avoid initial unnecessary movements. Setting the target velocity can be done by accessing 0x66 register. Setting the motor to velocity control mode can be done by accessing the 0x63 register. The actual velocity can be read by accessing the 0x6A register.

Register	Datagram	Description
0x54	0xD400640001	Torque loop P = 100, I = 1
0x56	0xD600640001	Flux loop P = 100, I = 1
0x5E	0xDE000061A8	Torque/Flux limit = 25000
0x60	0xE000000320	Velocity limit = 800 rpm
0x52	0xD200000003	Select encoder electrical angle source
0x50	0xD000000009	Velocity source selection/mechanical
0x58	0xD864000500	Velocity loop P = 25600, I = 1280
0x66	0xE600000064	velocity target = 100 rpm
0x63	0xE300000002	Activate velocity mode

Table 5 - Register Accessing in Velocity Mode

VII. POSITION MODE SETTINGS

The boot sequence listed in Table 6 will control the motor in position control mode. The motor will be held in the position 0 (position after rotor was aligned with the magnetic field for an incremental encoder) by loading this boot sequence. In position control mode, it is essential to have both velocity and position sources properly selected. Encoder mechanical angle and position was set as velocity and position sources by accessing 0x50 and 0x51 registers respectively. Position target can be set by accessing 0x68 register if the position is written to the chip manually. If the motor is driven by step and direction signals, step size per pulse input can be set by using 0x78 register. The direction of the rotation is defined by XORing the direction pulse and the sign of the step width input. After the rotor was initiated to the initial position, it is important to set the new rotor position to zero, by writing zero to the 0x6B register. The 0x6B register can be used to read the actual position during the motor operation. The motor should be set to the position control mode, after setting the target position value to avoid initial unnecessary movements. Setting the motor to position control mode can be done by accessing 0x63 register.

.

Register	Datagram	Description
0x54	0xD400640001	Torque loop P = 100, I = 1
0x56	0xD600640001	Flux loop P = 100, I = 1
0x5E	0xDE000061A8	Torque/Flux limit = 25000
0x60	0xE000000320	Velocity limit = 800 rpm
0x52	0xD200000003	Select encoder electrical angle source
0x50	0xD000000009	Velocity source selection/mechanical
0x58	0xD864000500	Velocity loop P = 25600, I = 1280
0x51	0xD100000009	Position source selection/mechanical
0x5A	0xDA08000000	Position loop P = 2048, I = 0
0x6B	0xEB00000000	Clear actual position
0x68	0xE800000000	Position target
0x78	0xF8000007D0	Position loop step size
0x63	0xE300000003	Activate position mode

Table 6 - Register Accessing in Position Mode

VIII. RESULTS

By programming the TMC4671-BOB using the commands stated in Section III to VII, the rotor is aligned with the stator magnet field and the motor is initialized in the desired operation mode. The performance of the motor is dependent on the characteristics of the motor as the optimum PI parameters depend on the motor characteristics. Hence, it is mandatory to tune the motor to obtain higher efficiency and accuracy. Fig. 5 and Fig. 6 shows the motor behavior in velocity control mode for tuned and un-tuned PI parameters respectively. Fig. 7 and Fig. 8 shows the motor behavior in position control mode for tuned and un-tuned PI parameters

respectively. The PI regulators were tuned by following the standard procedure described by Trinamic [9].

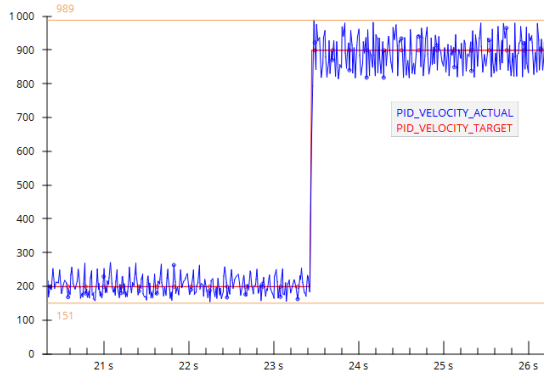

Fig. 5 – Velocity Mode Behavior for Tuned PI Parameters

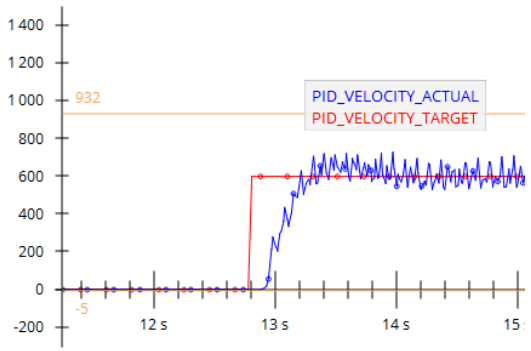

Fig. 6 - Velocity Mode Behavior for Un-tuned PI Parameters

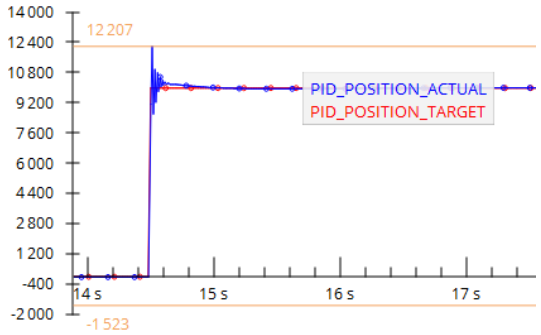

Fig. 7 – Position Mode Behavior for Tuned PI Parameters

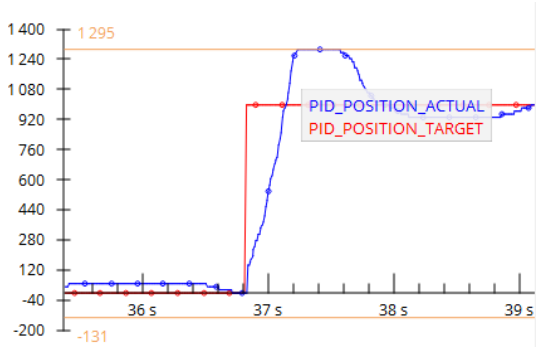

Fig. 8 - Position Mode Behavior for Un-tuned PI Parameters

IX. CONCLUTION

Unlike complex DSP solutions where the FOC algorithm needs to be implemented by writing firmware, the utilization of TMC4671 BOB is an easy and efficient solution as the FOC algorithm is already implemented in the chip. The user only needs to configure the registers of TMC4671-LA chip accordingly. The utilization of TMCL IDE and Trinamic RTMI adapter has made the motor tuning is an easy task where the system performance can be increased very efficiently.

REFERANCES

[1] A. Abdul Ali, F. Abdul Razak and N. Hayima, "*A Review on The AC Servo Motor Control Systems*", ELEKTRIKA- Journal of Electrical Engineering, vol. 19, no. 2, pp. 22-39, 2020. Available: 10.11113/elektrika.v19n2.214 [Accessed 16 December 2021].

[2] K. Hasse, "*Zur dynamik drehzahlgeregelter antriebe mit stromrichtergespeisten asynchron-kurzschlussläufermaschinen*", Ph.D, T. U. Darmstadt, 1969.

[3]. F. Blaschke. "*The principle of field-orientation as applied to the transvector closed loop control system for rotating-field machines: Siemens Rev.*", vol. 34, no. 1, pp. 217–220, 1972.

[4] *TMC4671 Datasheet*. Trinamic, 2021.

[5] *Implementation of a Speed Field Oriented Control of 3-phase PMSM Motor using TMS320F240*. Texas Instruments, 1999.

[6] D. Garcia, *Precision Digital Sine Wave Generation with the TMS32010*. Texas Instruments, 1989.

[7] *dsPIC33F Family Data Sheet.* Microchip Technology Inc., 2006.

[8] *TMC4671 BOB Description*. Trinamic, 2020.

[9] AN53: TMC4671 PI Tuning. Trinamic, 2020.