

RX62T

R01AN0899EU0201

Rev. 2.01

Jul. 30, 2014

Position Control of Permanent Magnet Synchronous Motors (PMSM) with Encoder

Introduction

This document presents the RX62T position control with a PMSM, which has been implemented on the RX62T evaluation kit with Hall sensors and encoder.

This document describes the hardware platform, methodology of position control, control block diagram, software structure, and flow chart of the position measurement and control.

The solution in this application note has been implemented with the RX62T evaluation kit and a 3-phase 8-pole 24V PMSM with a 1000 line single-ended encoder.

Target Device

RX62T

Contents

1. Overview	2
2. Specifications and Performance Data.....	3
3. Hardware Platform	4
4. RX62T Encoder Capture Function.....	5
5. Encoder Based Position and Speed Calculation	8
6. Position Control Strategy	11
7. Software Description	15
8. Motor and Position Control Parameters.....	19
9. Demonstration Guide	21
Appendix A - References	33

1. Overview

Position control plays an important role in various areas such as automation industry, semiconductor industry, etc. PMSMs are ideal for advanced position control systems, for their potentials of high-efficiency, high torque to current ratio, and low inertia, and PMSMs have been widely used in the industrial fields. Various approaches have been made to realize high-performance motion control.

With successively improving reliability and performance of digital controllers, advancements in microcontrollers (MCU) have greatly enhanced the potential of PMSMs in servo position control applications. Digital control can be implemented by MCUs. An MCU is much more compact, reliable, and flexible which makes it superior to an analog-based stepper control. High-performance of PMSM can be obtained by means of field-oriented control, which is only realizable in a digital based system.

The RX62T is a 32-bit high-performance microcontroller with a maximum operating frequency of 100MHz and 165 DMIPS and single precision floating-point unit (FPU), which is equipped with multifunction timers (MTU, GPT), high-speed 12-bit A/D converter and encoder signal capture for facilitating servo motion control.

In this application note, a RX62T FPU based position motion control system is proposed. Position regulation is developed to provide both a trajectory generator and a PID controller, which ensures accurate position control and fast tracking. The trajectory generator provides position set-point commands, and the position PID controller operates on the position error and outputs a current command. The current regulation with field-oriented control is implemented to secure fast dynamic response.

The software described in this application note is applicable to following devices and platforms:

- ❖ MCU: RX62T and RX62N
- ❖ Motor: Three-Phase PMSM
- ❖ Platform: Renesas Evaluation Kit
- ❖ Control Algorithm: Encoder Based Position Control

2. Specifications and Performance Data

The implementation of position control is based on the Renesas evaluation kit and the RX62T MCU. The main specifications are described as following:

- ❖ Input Voltage: 24VDC
- ❖ Rated Bus Voltage: 24V
- ❖ Output Voltage: 24VAC
- ❖ Rated Output Power: 120W
- ❖ PWM Switch Frequency: 20KHz
- ❖ Control Loop Frequency: 10KHz
- ❖ Current Measurement: 3 Shunt Resistors
- ❖ Position Measurement: 1000 Line Quadrature Encoder
- ❖ Implementation: FPU
- ❖ CPU Bandwidth: 17%
- ❖ Used Flash Memory: 13.444Kbytes
- ❖ Used RAM: 1.725Kbytes
- ❖ Used Stack : 336bytes

3. Hardware Platform

RX62T FPU based position control is implemented with the Renesas RX62T evaluation kit and a three-phase PMSM with a 1000 line single-ended encoder as shown in Figure 1.

RX62T evaluation kit is a single board inverter, based on the RX series microcontroller RX62T.

The board has the following features:

- ❖ A complete 3-phase inverter on-board with a low-voltage motor
- ❖ 24V external power supply to provide DC bus voltage, 15V and 5V power supply
- ❖ Power devices use Renesas low voltage MOSFETs
- ❖ Power rate up to 120 watts
- ❖ Supports three-shunt and single-shunt current measurements
- ❖ Easily change jumpers from the external amplifiers to the internal Programmable Gain Amplifier (PGA)
- ❖ USB communication with the PC via a H8S2212 MCU
- ❖ Graphic User Interface (GUI) used to both modify the motor and control parameters and tune the speed and position control
- ❖ Connectors for Hall sensors and encoder connections
- ❖ LCD to monitor the operation status
- ❖ Supports the standalone mode set by potentiometer and push buttons
- ❖ Supports the second motor drive, signals and connector for another motor control power stage are available

The motor is a 24V 4 pair poles 3-phase PMSM with

- ❖ 3 Hall sensors
- ❖ 1000 line quadrature encoder

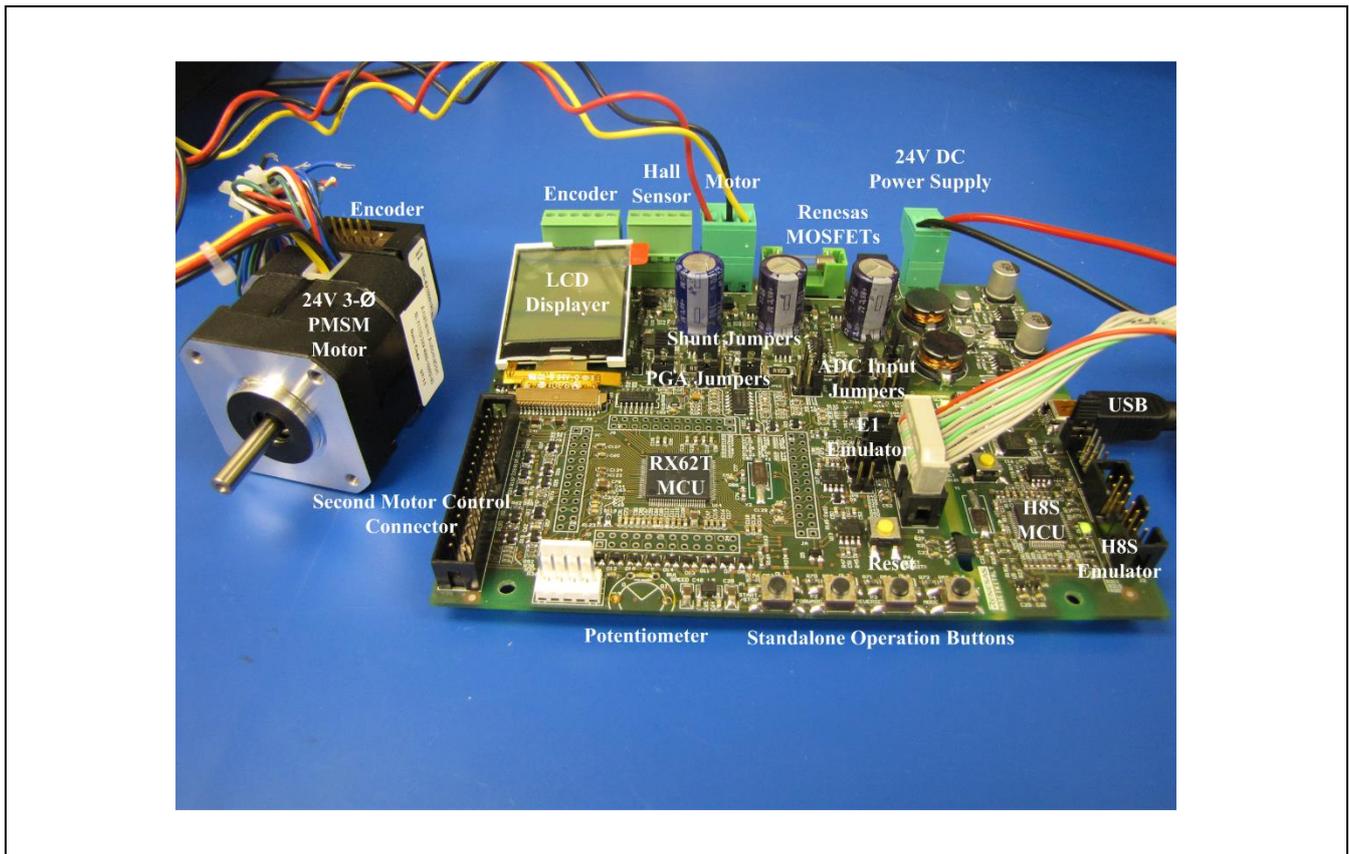


Figure 1 System Hardware Setup (Motor and Control Platform)

4. RX62T Encoder Capture Function

The RX62T is a 32-bit high-performance microcontroller with a maximum operating frequency of 100MHz, 165 DMIPS, and single precision FPU, which is equipped with multifunction timers (MTU, GPT), high-speed 12-bit A/D converter, and 10-bit A/D converter for facilitating motor control. Figure 2 shows the block diagram of a sensorless vector control of PMSM based on the Renesas RX62T Microcontroller.

The RX62T has a dedicated function for the encoder measurement as depicted in Figure 2. The MTU3 timer external clock input TCLKA, TCLKB, TCLKC, and TCLKD can be used for two-phase encoder pulse inputs. When the MTU3 timer of Channels 1 and 2 are specified by the phase counting mode an external encoder clock is selected as the counter input clock, and the TCNT operates as an up/down-counter. The phase difference between two external input clocks is detected, and the TCNT is incremented or decremented accordingly. The rotor position and speed can be measured by reading the TCNT counts.

The following summarizes the MTU3 function for the encoder pulse counting functionality:

- ❖ MTU Channel 1 & 2 support 2-phase pulse counting mode which is called “Phase Counting Mode”
- ❖ This function covers 4 modes
- ❖ At these modes, the counter works as up/down counter, and it is possible to detect the direction of counter with the flag.
- ❖ Up/down count by detecting phase difference between phase A and B of the encoder on mode 1 and mode 4
 - Mode 1: every rising edge & falling edge of both phases of the encoder pulse
 - Mode 4: every rising edge & falling edge of phase B encoder pluse
- ❖ Up/down count by two pulse lines which indicate the direction, speed and position.
 - Mode 2: One pulse line and one direction
 - Mode 3: Two pulse lines for each direction
- ❖ The MTU can detect automatically speed and position data as the pulse width & the pulse. The data of speed and position can be captured every periodic cycle.

In this application, the encoder pulses of A and B are the inputs to the TCLKA and TCLKB. The Z pulse is the input to IRQ0. For the second motor, the encoder pulses of A and B are the inputs to the TCLKC and TCLKD. The Z pulse is the input to IRQ3.

The host communication using the GUI is communicated with the RX62T MCU by the USB communication. It can display the motor operation status in the real time, modify the motor and control parameters, and tune the speed and position control.

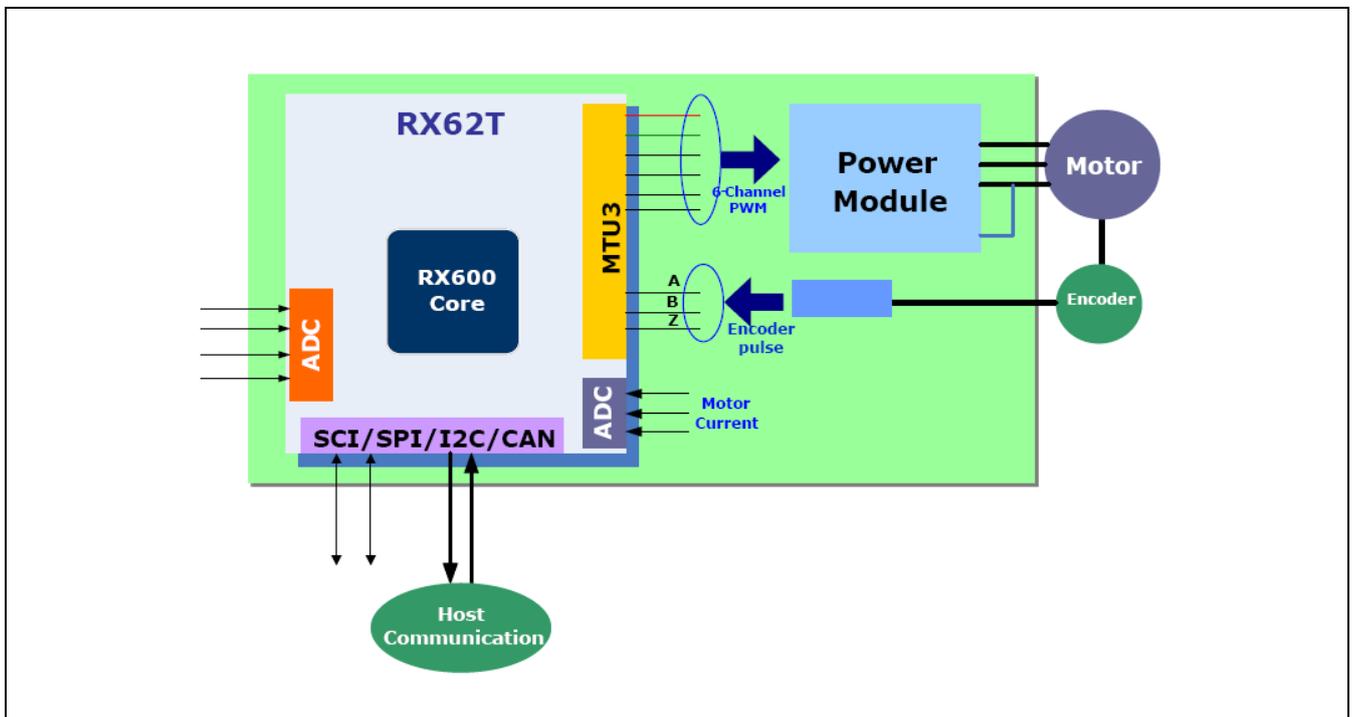


Figure 2 RX62T Encoder Capture Functionality

Table 1 lists the timer register function for Channel 0 to 2 for the encoder capture. The timer MTU is enabled to automatically detect both the pulse-width and the number of pulses from the encoder every speed control loop period. External wiring is not necessary for any trigger signals. The encoder signals are directly input to the timer external clock. TCLKA and TCLKB are clock sources of channel 1 and input command pulses to the timer external clock. TCLKC and TCLKD are clock sources of channel 2.

- ❖ Channel 1 counter is counted by every falling edge and rising edge of the encoder pulse.
- ❖ Channel 0 is used for interval time to generate input capture trigger of Channel 1 and Channel 2 and the interrupt of speed control loop.
- ❖ Channel 2 measures the pulse command input.
- ❖ Channel 0 compare match (speed control loop period) can be selected as an input capture trigger for Channel 1 internally.
- ❖ Channel 1 and Channel 2 external timer clocks (encoder pulse or command pulse) can be selected as input capture triggers for Channel 0 internally.

Table 1 MTU Timer Registers Function

Ch	Register/Counter	Function	Data
0	TCNT0 (Ch0 Counter)	Free Running Timer by internal clock	Interval timer
	TGR0A	Input capture register (trigger by every edge of encoder pulse which is used for ch1 counter clock)	Pulse width of encoder pulse
	TGR0B	Output compare register (to create input capture trigger for Ch1 & Ch2)	Speed Control Loop Period
	TGR0C	Buffer register of TGRA0	Last data of pulse width
	TGR0D	Output compare register (to create input capture trigger for Ch1 & Ch2)	Position Control Loop Period
1	TCNT1 (Ch1 Counter)	Up/Down Counter by encoder pulse	Up/Down Counter shows position and speed
	TGR1A	Input capture register (trigger by Ch0/TGR0B compare match)	The number of pulse of encoder at every speed control loop period
	TGR1B	Input capture register (trigger by Ch0/TGR0D compare match)	The number of pulse of encoder at every position control loop period
2	TCNT2 (Ch2 Counter)	Up/Down Counter by command input	Up/Down counter shows command of speed and position
	TGR2A	Input capture register (trigger by Ch0/TGR0B compare match)	The number of pulse of command at every speed control loop period
	TGR2B	Input capture register (trigger by Ch0/TGR0D compare match)	The number of pulse of command at every position control loop period

Figure 3 shows how the MTU captures the encoder signals in phase counting mode. The Channel 1 is coupled with Channel 0 to input 2-phase encoder pulses of a servo motor in order to detect position or speed. Channel 1 is set to phase counting mode 1, and the encoder pulse A-phase and B-phase are input to MTCLKA and MTCLKB. In Channel 0, MTU3_0.TGRC compare match is specified as the TCNT clearing source and MTU3_0.TGRA and MTU3_0.TGRC are used for the compare match function and are set with the speed control cycle and position control cycle. MTU3_0.TGRB is used for input capture, with MTU3_0.TGRB and MTU3_0.TGRD operating in buffer mode. The Channel 1 counter input clock is designated as the MTU3_0.TGRB input capture source, and the widths of 2-phase encoder 4-multiplication pulses are detected. MTU3_1.TGRA and MTU3_1.TGRB for Channel 1 are designated for the input capture function. MTU3_0.TGRA and MTU3_0.TGRC compare matches in Channel 0 and are selected as the input capture sources to store the up/down-counter values for the control cycles.

Therefore, the RX62T MTU itself can realize precise detection of the pulse-width and the number of pulses, which are needed to estimate motor speed and position, and it minimizes the load of the CPU to detect those. Also, the MTU is able to receive the pulse command as well.

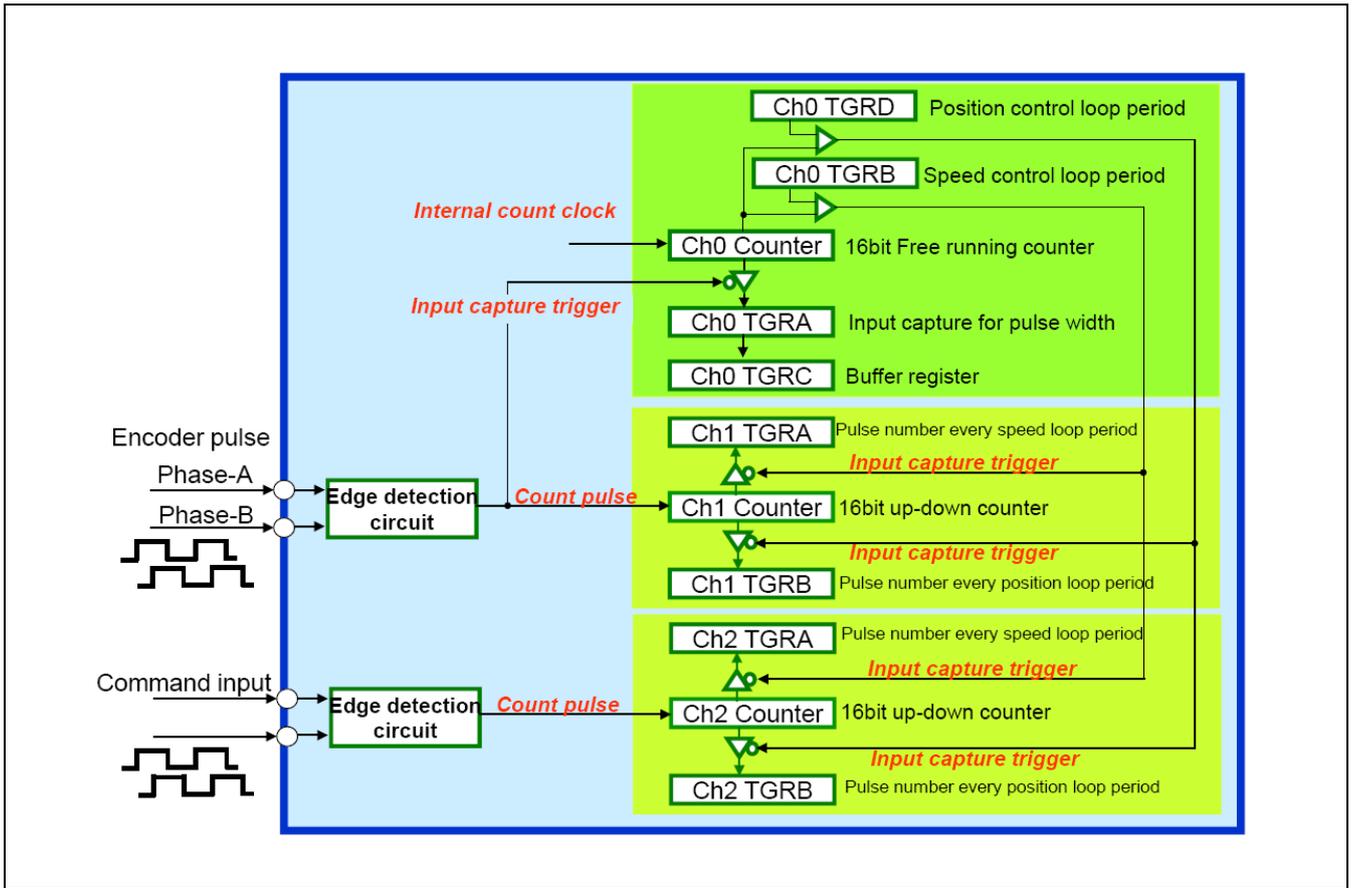


Figure 3 Encoder Pulse Capture in Phase Counting Mode

5. Encoder Based Position and Speed Calculation

5.1 Position and Speed Measurement

A digital encoder outputs three pulse trains: A, B and Z, as shown in Figure 4. These pulses are fed into timer units TCLKA and TCLKB that count events. Pulses A and B are offset by 1/4th of the distance to give a 90-degree offset, so they are known as quadrature counts. Pulse Z occurs only once per rotation, and it is fed into the interrupt input (IRQ0) and zeroes out (resets) the counter MTU2_TCNT. When the pulse Z occurs, the rotor angle with respect to the stator frame produces a definite value, preferably zero. If this value is not zero, it is a constant offset that can be measured. Quadrature counters are designed to count these pulses up or down, depending on whether A comes before or after B. That is, the relationship between A and B indicates the direction of rotation.

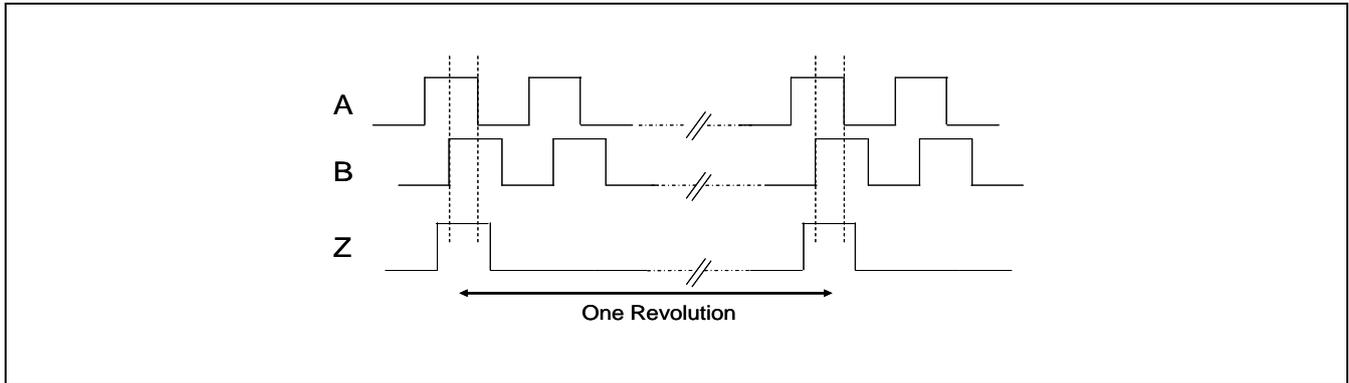


Figure 4 Relationship Among the Digital Encoder Pulses A, B and Z

The encoder has been aligned and calibrated with Hall sensor U with zero initial position. The angle is zero count when the Z pulse occurs through the external interrupt IRQ0. From this point onwards it is given a certain count value as the quadrature counter is read. As shown in Figure 5, the phase counting mode 1 is used to up/down count by detecting phase difference between A and B phase. These counts are transformed into a proper angle value for the rotor position.

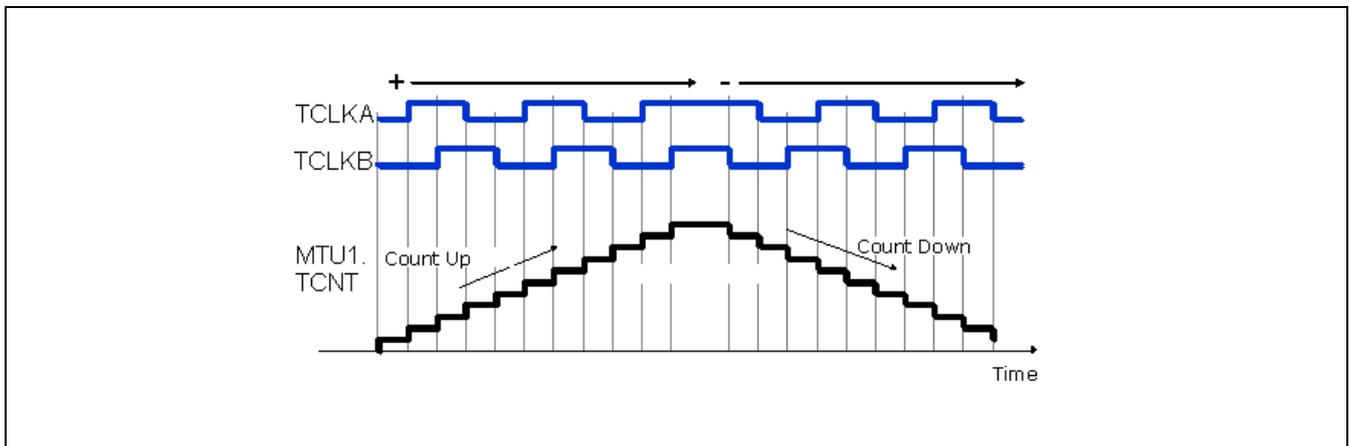


Figure 5 Encoder Counting Mode Operation

Motor speed determines how much the angle of the rotor changes over time. As shown in Figure 6, pulses A and B from the encoder are used at the control loop rate. Two angles are measured at constant time intervals, thus giving the measurements needed to compute speed: delta angle and delta time. Speed is computed by dividing the delta angle, $\Delta\theta$, by the delta time.

The motor position is the number of the encoder pulse as $N(m+1) - N(m)$ where m is the number of consecutive samples.

$$\Delta\theta = N(m+1) - N(m)$$

and the motor speed is

$$\omega_r = \frac{N(m+1) - N(m)}{T_{sp}}$$

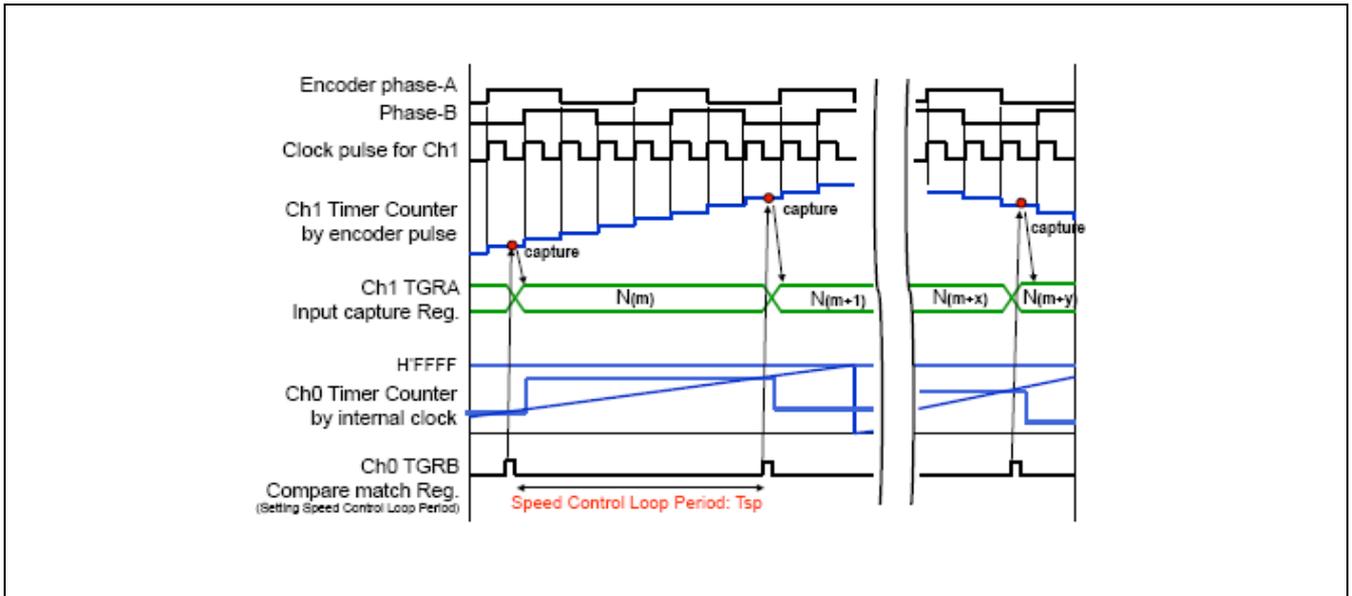


Figure 6 Speed Calculation Using Encoder Pulses A and B at Control Loop Rate

5.2 Initial Position Identification

Incremental encoders can only give displacements from the initial position and can't provide absolute position. For PMSM and position control, the initial position is required. Although alignment has been calibrated, the initial starting position before the Z pulse is still unknown.

By means of Hall sensors the rotor initial position can be identified and further corrected when the rotor starts rotating. Assuming the Hall sensors are located at each phase, as shown in Figure 7. The output signals of the Hall sensors are illustrated in Figure 8. It can be seen that the resolution of the Hall sensor signals are 60° (electrical degree). Table 2 shows the possible combinations corresponding to different positions.

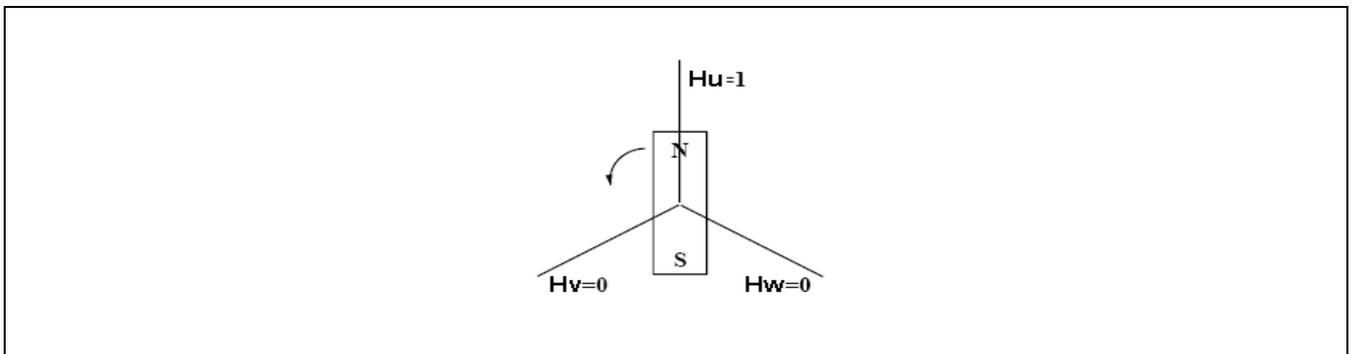


Figure 7 Hall Sensors for Initial Rotor Position

From Figure 8 and Table 2, given a specific Hall sensor output combination, the rotor must reside in certain section with a range of 60° . The initial position is determined as follows. When a group of output signals are obtained, for example, (101), we can decide which section the rotor is in (section 1 in this example). We can set the initial position at the center of the section (30° in this example). It can be seen that the maximum error of the initial position is 30° , which occurs when the rotor is at the edge of two regions. However, even with 30° error, the motor is still able to produce sufficient torque to start the motor.

Once the motor starts rotating, the position can be readily corrected when the rotor moves out of the initial section and enters the next section. This position is accurate. In the previous example, when the motor starts rotating in the positive direction from section 1, the rotor position can be corrected when the position is 60° .

Table 2 Relationship Between Hall Sensors and Rotor Position

Section	Hu	Hv	Hw	Rotor Position
1	1	0	1	0~60
2	1	0	0	60~120
3	1	1	0	120~180
4	0	1	0	180~240
5	0	1	1	240~300
6	0	0	1	300~360

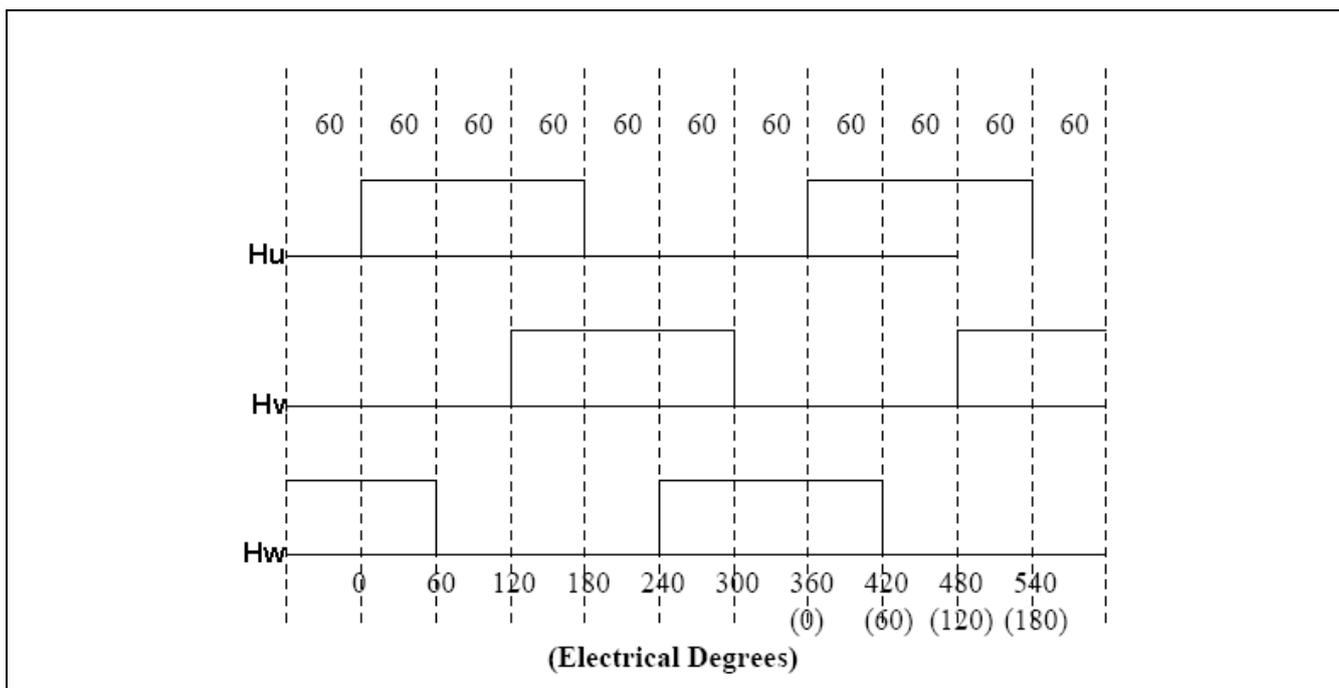


Figure 8 Hall Sensor Output Signals

6. Position Control Strategy

6.1 Block Diagram of Position Control

Figure 9 is the block diagram of position control. The position control developed includes two loops. The outer loop is position control to make the motor track and hold the given position. The inner loop is current control. Actually, it is the torque control loop. The motor currents are sampled through three shunt resistors and converted into the dq axis currents. The control loop here is to control the q axis current for the torque.

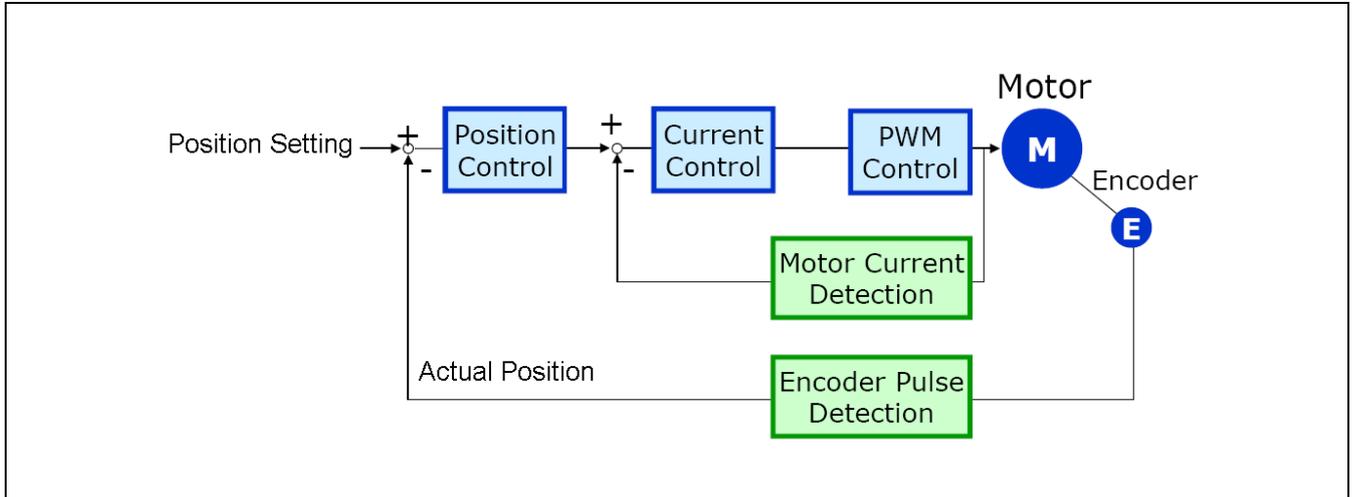


Figure 9 Block Diagram of Position Control

The position control scheme of the PMSM is illustrated in Figure 10. The system has an inner loop of current regulation using vector control, and an outer loop of position regulation. This dual-loop structure ensures the fast torque response by using the vector control, high position accuracy, and fast tracking performance with the position controller.

In order to determine the d and q axis currents, the phase currents must be measured. Vector formulation uses Clarke and Park transforms to convert the measured phase currents from the u, v, and w frame to first transform them in the static orthogonal (α, β) frame (which is 90 degrees apart), and then to the rotor frame. The rotor frame is also an orthogonal frame aligned along the magnetic field axes known as the d and q frame. These transformations use the transcendental functions sine and cosine of the rotor angle. Therefore, it is a requirement that the rotor angle is known at the time of the calculation. The position control requires current sensors and an encoder attached to the rotor shaft to measure the rotor position.

Once the currents are transformed into the d and q frame, the control algorithm simply runs the PID or PI loop to calculate the required voltages for the torque and flux. These required voltages (V_{dc} , V_{qc}) are then transformed back into the u, v, and w frame using the inverse Clarke and inverse Park transforms to further calculate the PWM duty cycle.

The position command is an input to the position control system. The motor has an encoder mounted on its rotor to give the quadrature pulses A and B, as well as the zero synch pulse Z. All three of the rotor position signals are sent to the MCU's input-capture and timer/quadrature counter peripheral for making position and speed measurements. The commanded position is compared with the actual rotor position. The position regulator uses the traditional PID controller, and outputs the torque control command of i_q^* to make the motor move and track the commanded position.

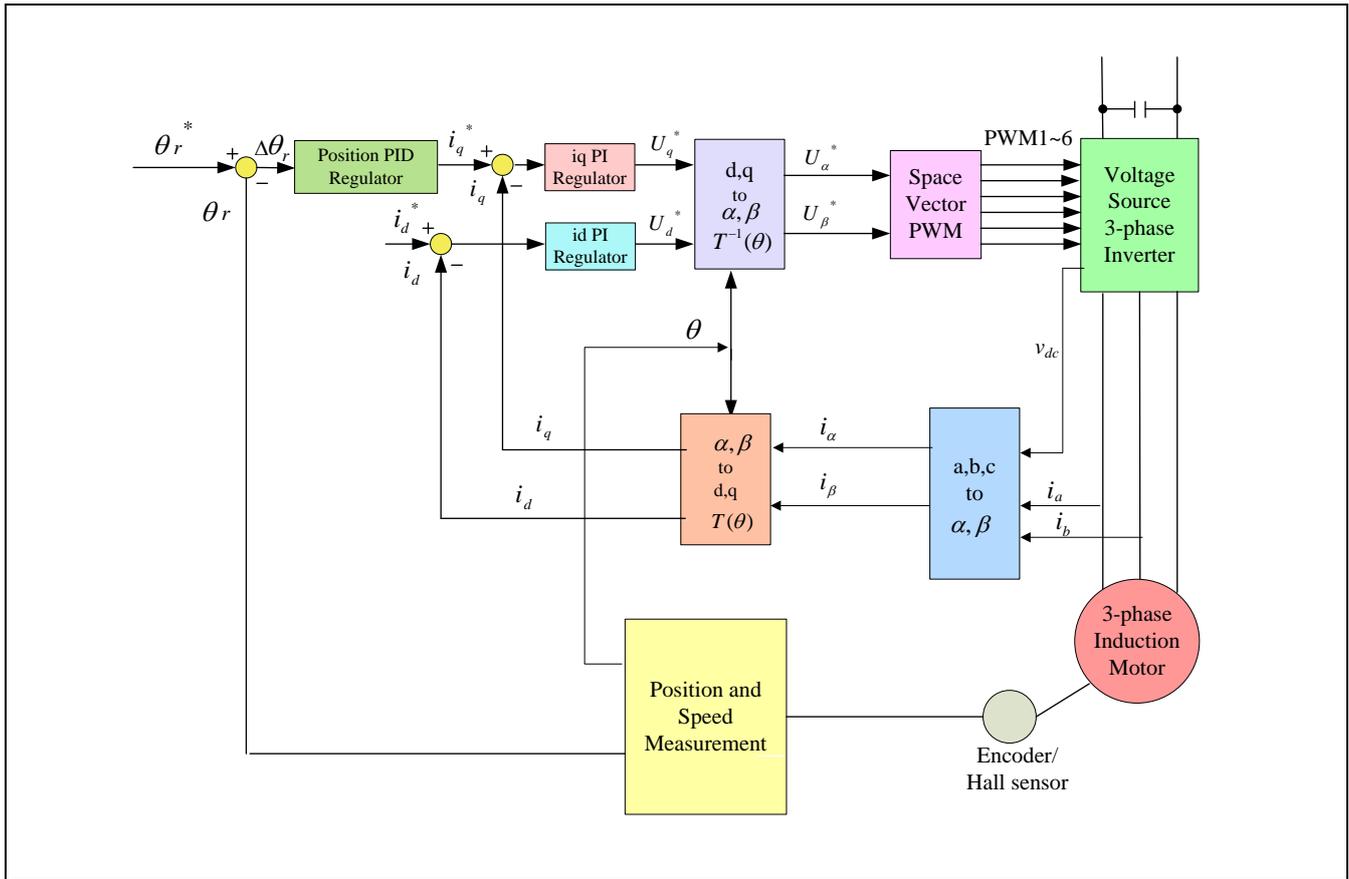


Figure 10 Position Control Scheme Diagram

6.2 Position Control Loop Design

The basic components of a typical servo position control system are depicted in Figure 11. In this figure, the servo position control closes a current loop as described in next section and is modeled simply as a linear transfer function $G_{ireg}(s)$. Of course, the servo drive has peak current limits, so this linear model is not entirely accurate. However, it does provide a reasonable representation for analysis. For the purposes of this discussion the transfer function of the current regulator, or really the torque regulator, can be approximated as unity for the relatively lower motion frequencies.

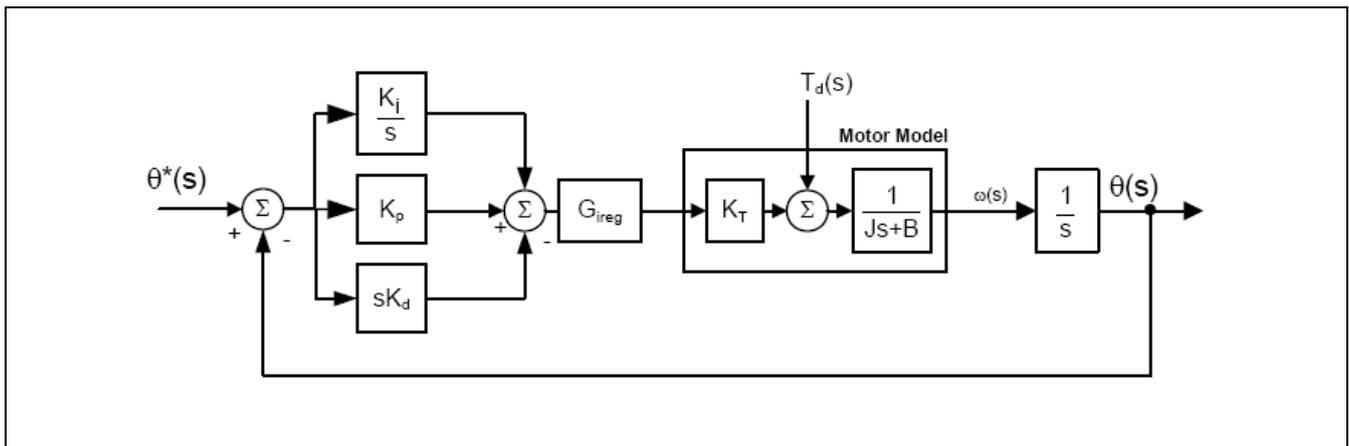


Figure 11 Position PID Controller Topology

The PMSM is modeled as a lump inertia, J , a viscous damping term, B , and a torque constant, K_t . The lump inertia term is comprised of both the servo motor and load inertia. It is also assumed that the load is rigidly coupled such that the torsional rigidity moves the natural mechanical resonance point well out beyond the position controller's bandwidth. This assumption allows us to model the total system inertia as the sum of the motor and load inertia for the frequencies that can be controlled.

An encoder coupled directly to the motor shaft measures the actual motor position, θ (s). External shaft torque disturbances, T_d , are added to the torque generated by the motor's current to give the torque available to accelerate the total inertia, J .

Around the current regulator motor block is the servo position controller that closes the position loop. The basic servo position controller provides both a trajectory generator and a PID controller. The trajectory generator provides only position set-point commands labeled in Figure 11 as $\theta^*(s)$. The PID controller operates on the position error and outputs a current command.

There are three gains to adjust in the PID controller, K_p , K_i and K_d . These gains all act on the position error defined as:

$$\Delta\theta = \theta^*(s) - \theta(s)$$

Note the superscript “*” refers to a commanded value.

The output of the PID is given mathematically in the time domain as:

$$i_q^*(t) = K_p \Delta\theta(t) - K_i \int \Delta\theta(t) dt + K_d \frac{d}{dt} \Delta\theta(t)$$

In general, the proportional term affects the overall response of the system to a position error. The integral term is needed to force the steady state position error to zero for a constant position command, and the derivative term is needed to provide a damping action as the response becomes oscillatory. Unfortunately, all three parameters are inter-related, so adjusting one parameter will affect any of a previous parameter adjustments.

Tuning the PID controller can be done if the motor and load parameters and the desired frequency response are known. They are adjusted using the parameters in the header file “customize.h”.

6.3 Current Control Loop

The current loop is a standard PI type based on the standard Park-Clarke stationary reference frame to rotary reference transformations. The initial rotor position is determined by the Hall sensors. Once a Hall transition occurs, the rotor position is then determined by reading the incremental encoder. The basic block diagram for the current vector control is shown in Figure 12.

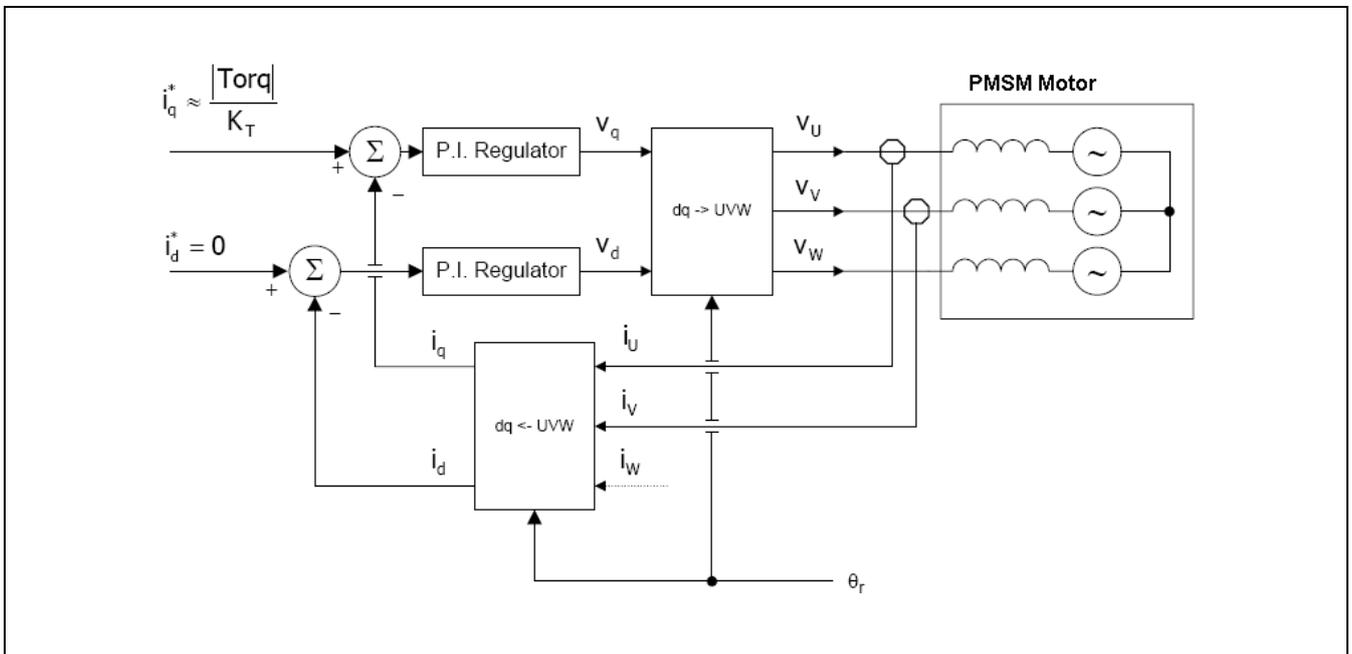


Figure 12 Block Diagram of Current Vector Control

Neglecting motor saliency, the commanded q axis current, i_q^* , is linearly related to the commanded torque. The “d” axis current command, i_d^* , is set to zero as field weakening is not required. The transformation takes two steps. First, the stationary currents are transformed to an arbitrary stationary pair of orthogonal axes (α, β). Second, the axes are then rotated to the rotor axes for control purposes.

The typical current PI controller is depicted in Figure 13. K_p and K_i are the proportional gain and integration gain, and they can be independently adjusted by the software. The hardware gain K_b takes into account the bus voltage.

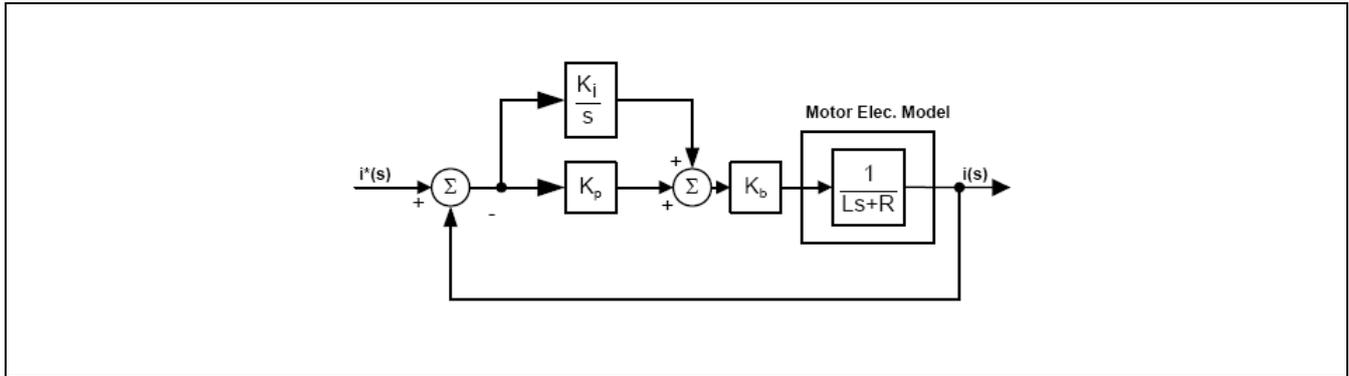


Figure 13 Current PI Controller Topology

The transfer function of the block diagram is:

$$\frac{i(s)}{i^*(s)} = \frac{\left(\frac{K_p K_b}{L}\right)s + \left(\frac{K_i K_b}{L}\right)}{s^2 + \left(\frac{K_p K_b + R}{L}\right)s + \left(\frac{K_i K_b}{L}\right)}$$

It has a characteristic equation in the form of:

$$s^2 + 2\xi\omega_0 s + \omega_0^2 = 0$$

Therefore:

$$K_p = \frac{2L\xi\omega_0 - R}{K_b}$$

$$K_i = \frac{\omega_0^2 L}{K_b}$$

The system exhibits the standard second order response with the addition of a real zero. To tune the system, the high frequency of 500Hz needs to be first set for K_p , and then slowly increase the integral term K_i to bring the steady state error to zero.

7. Software Description

7.1 Overall Software Structure

Position control algorithm is implemented with the complete C code using the Renesas RX62T MCU FPU. The overall software structure is shown in Figure 14.

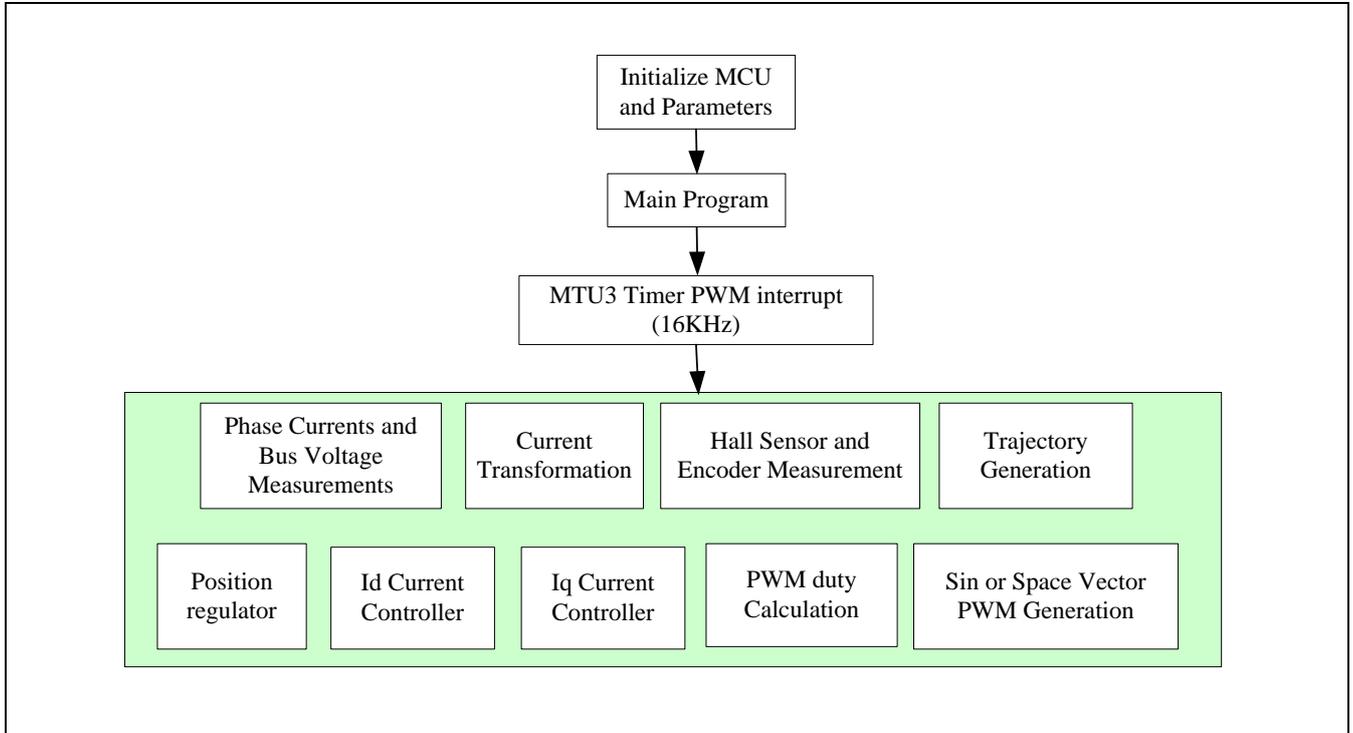


Figure 14 Position Software Architecture

The procedures include:

- ❖ Initializations of RX62T MCU Motor and Control Parameters
- ❖ Current Offset Calculations
- ❖ Phase Currents and Bus Voltage Measurements
- ❖ Hall Sensor and Encoder Measurements
- ❖ Initial Position Identification
- ❖ Rotor Position Calculations
- ❖ Vector Control Transformations
- ❖ Motion Profile: Trajectory Generation
- ❖ Position Regulator
- ❖ Current Controllers
- ❖ PWM Duty Calculation
- ❖ Space vector PWM generation

7.2 Software e²studio Workspace

Shown in Figure 15 is the workspace for position control using Renesas' e²studio IDE (Integrated Development Environment).

- ❖ All code is written in the floating point C language;
- ❖ The software is modularized according to the position control block diagram (as shown in Figure 10);
- ❖ I/O definitions and basic MCU drivers are automatically ported by e²studio;
- ❖ Motor and control parameters are easily tuned through the header file "customize.h" and the GUI.

The code includes: dbset.c; hwsetup.c; intprg.c; main.c; mcrlibf.c; motorcontrol.c; resetprg.c; userif.c and vectbl.c.

- ❖ dbset.c includes structures used by the runtime library both to clear un-initialized global variables and to write initial values into initialized global variable sections.
- ❖ hwsetup.c is the hardware initializations.

- ❖ vecttbl.c contains the array of addresses of ISRs.
- ❖ resetpr.c has functions called just after reset.
- ❖ intprg.c is entry points for all of standard ISRs vectors.
- ❖ main.c includes: initialization of control parameters, MTU3 timer, interrupts, serial communication, encoder capture definitions; and uploading eeprom parameters. The current sensor offsets are calculated before the output of the PWMs. The while loop executes parameter updates and SCI communication with the GUI.
- ❖ The motorcontrol.c is the major code for position control, which contains most of functions and function calls to implement position control.
- ❖ Mcrplib.c includes vector control transformations – Clarke, Park, and inverse Clarke and Park transformations, and sine and space vector PWM generation.

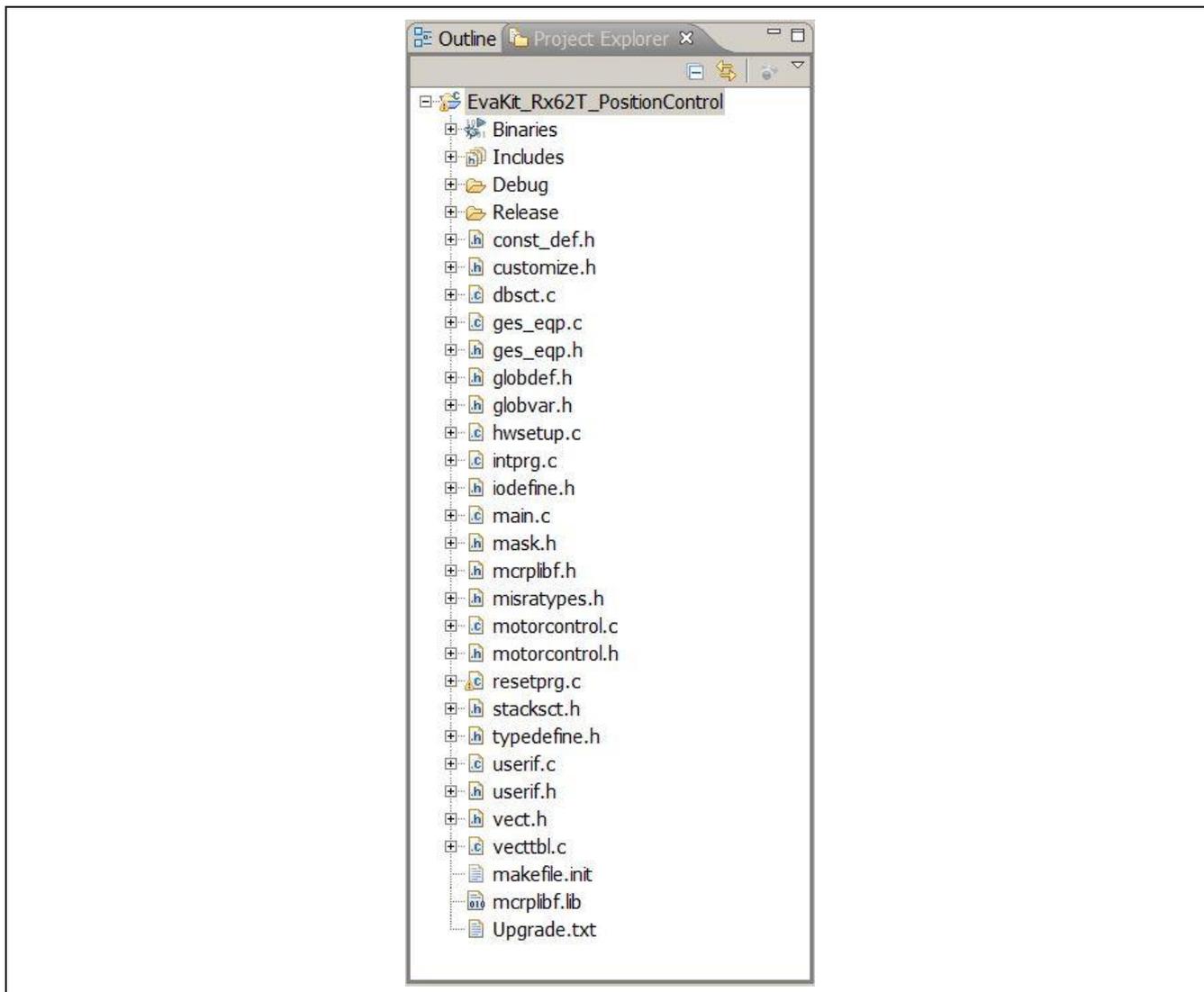


Figure 15 Encoder Counting Mode Operation Position Control Software Workspace

7.3 Hall and Encoder Based Position and Speed Measurement

Figure 16 is a flowchart of position measurement. The procedures for the position measurement based on Hall sensors and encoder are:

- ❖ Initialize Hall sensor, encoder capture timer registers, and I/O ports;
- ❖ Identify the rotor initial position using Hall sensor;
- ❖ Move the motor to capture the position using encoder pulses;
- ❖ Calibrate the rotor position once the Hall commutation changes;
- ❖ After calibration, recalculate the rotor position;
- ❖ Check encoder Z pulse and reset the position offset and encoder pulse capture timer count;
- ❖ Calculate the rotor position and motor speed.

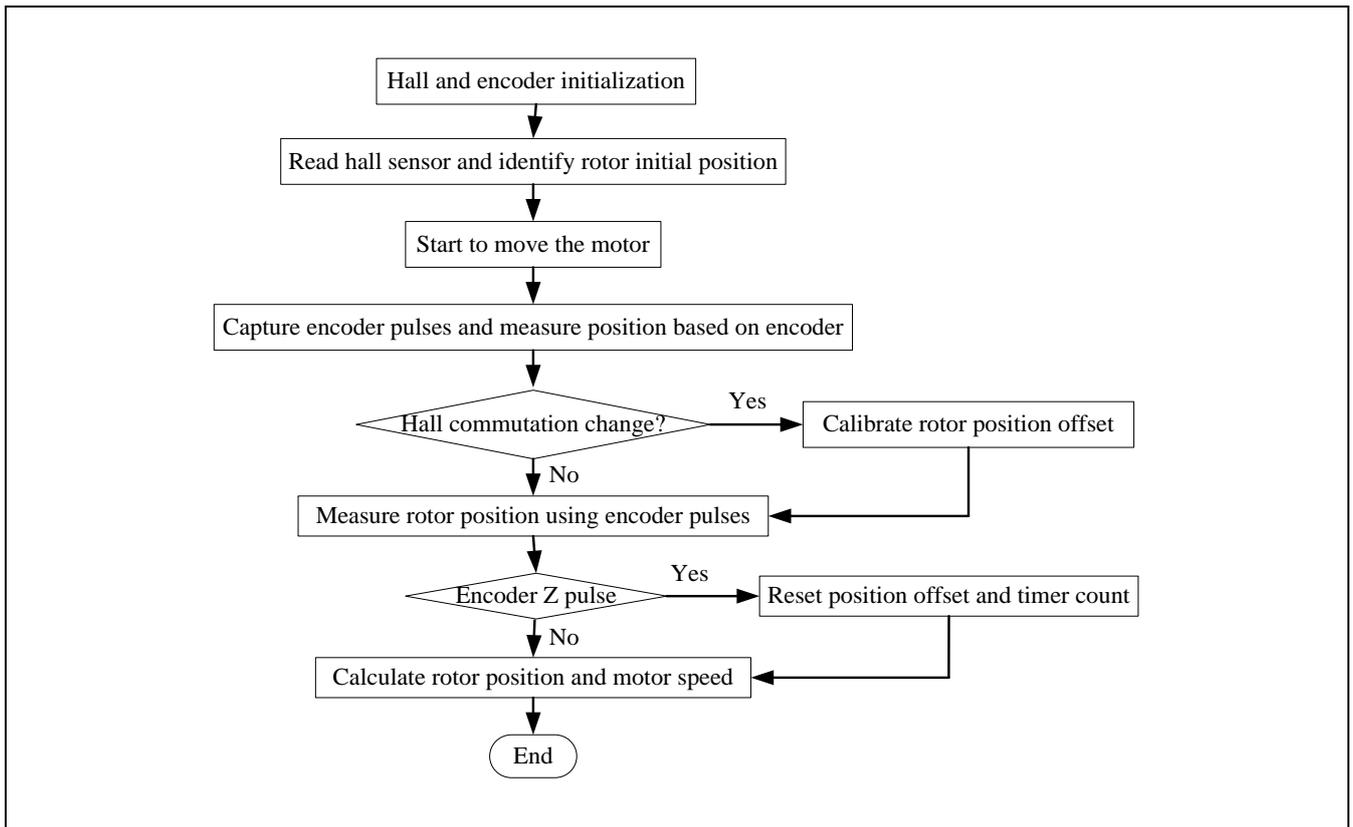


Figure 16 Encoder Counting Mode Operation Flowchart of Position and Speed Measurement

7.4 PWM Interrupt for Position Control

The position profile generation and position control are put in the PWM interrupt with a 16 kHz carrier frequency. Figure 17 is a flowchart of PWM interrupt.

The procedures in the PWM interrupt of MC_ConInt () are:

- ❖ Measure motor phase currents and DC bus voltage;
- ❖ Calculate motor position and speed using Hall sensors and encoder;
- ❖ Transfer motor currents into dq currents;
- ❖ Current control loop;
- ❖ Update trajectory generator and position profile;
- ❖ Position control loop;
- ❖ PWM generation using space vector PWM modulation or sinusoidal PWM modulation.

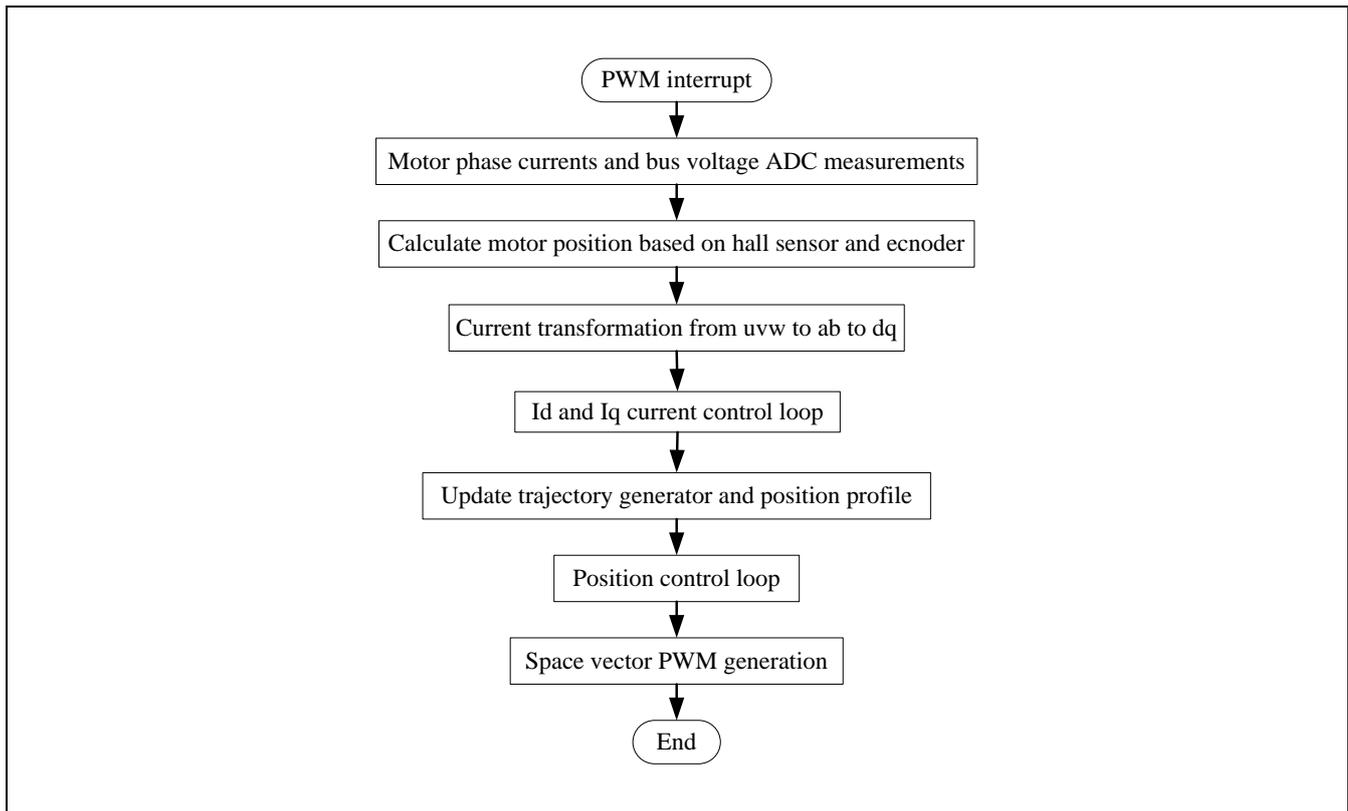


Figure 17 Flowchart of PWM Interrupt for Position Control

8. Motor and Position Control Parameters

8.1 Tuning Through the Header File

According to the motor datasheet and position control requirements, the motor and control parameters and the motion profile should be properly tuned.

Motor and control parameters required in the header file “customize.h” include:

❖ #define	ENC_EDGES_CUSTOM	4000	// Total encoder Edges/Revolution
❖ #define	PWM_FREQ_CUSTOM	16000	// PWM Frequency in Hz
❖ #define	SAM_FREQ_CUSTOM	16000	// Sample Frequency in Hz
❖ #define	C_POLI_CUSTOM	4	// Polar couples number
❖ #define	R_STA_CUSTOM	8	// Stator phase resistance in Ohm/OHM_DIV
❖ #define	L_SYN_CUSTOM	10	// Synchronous inductance in Henry/HEN_DIV
❖ #define	POS_MIN_CUSTOM	0	// Minimum position in counts
❖ #define	POS_MAX_CUSTOM	40000	// Maximum position in counts
❖ #define	KP_CUR_CUSTOM	60	// K prop. current control
❖ #define	KI_CUR_CUSTOM	80	// K integ. current control
❖ #define	K_P_POSITION	10	// K prop. position control
❖ #define	K_I_POSITION	12	// K integ. position control
❖ #define	K_D_POSITION	150	// K derivative position control

8.2 Operation Using the GUI

The motor and control parameters can be tuned with the Renesas GUI as shown in Figure 18. Without modifying the code, the parameters can be set or changed for different motors and applications. There is a parameter window to set up 20 separate parameters. By scrolling up and down through these parameters, the user can decide to make changes to the settings, and “Write” to EEPROM, but this doesn’t change the “customize.h” file. The original values will be restored upon clicking “Reload”. From Figure 19, it can be seen that these parameters mirror the #defines in the “customize.h” file, and the motor and control parameters can be easily changed using the GUI.

The GUI has a position control window to set the commanded position and display the motor actual operation status.



Figure 18 GUI Interface of Evaluation Kit

INDEX	DESCRIPTION	UNIT	MIN	MAX	VALUE	VALID
1	00. Default Parameters Setting	-	0	32767	0	true
2	01. Minimum Speed	rpm	200	5000	500	true
3	02. Maximum Speed	rpm	1000	20000	2500	true
4	03. Acceleration	rpm/s	1	10000	1000	true
5	04. Deceleration	rpm/s	1	10000	1000	true
6	05. Polar couples	-	1	5	5	true
7	06. Startup Current	Apk/10	0	5000	10	true
8	07. Maximum "q" Current	Apk/10	0	5000	20	true
9	08. Stator Resistance	Ohm/10	0	5000	17	true
10	09. Synchronous Inductance	Henry/10000	0	5000	12	true
11	10. Startup Time	ms	300	10000	1000	true
12	11. Current Loop Kp	-	0	2047	60	true
13	12. Current Loop Ki	-	0	1023	80	true
14	13. Speed Loop Kp	-	0	4095	10	true
15	14. Speed Loop Ki	-	0	4095	100	true
16	15. Startup offset V	V/10	0	32767	0	true
17	16. Startup delta V	V/10	0	32767	0	true
18	17. PI Tuning trigger	-	0	32767	0	true
19	18. Free	-	0	32767	0	true
20	19. Free	-	0	32767	0	true

Figure 19 Parameter Window

9. Demonstration Guide

9.1 Introduction to the Demonstration Guide

The purpose of this Demonstration Guide is to help users get up and running quickly with the RX62T motor control kit (YMCRPRX62T). The RX62T Microcontroller is pre-programmed to run “Three-Shunt Sensorless Vector Control with External Amplifier”. Therefore, the user does not need to reprogram the board using the E1 programmer/debugger to demonstrate the Position Control of PMSM with Encoder, and later sections will explain how to (1) setup the demo board, (2) build and debug the demo project with e²studio, and (3) run the GUI application. The user needs to connect the motor and the power supply to experience the efficient motor control capabilities of the Renesas RX62T microcontroller.

Caution: Do not connect power to the board until all instructions are followed.

The Demo contains the following items:

- RX62T Motor Control Evaluation Board (YMCRPRX62T)
- One BLDC Motor with a 3-way Molex connector and encoder cable
- 24V DC Power Supply
- Mini-USB cable
- CD ROM for motor firmware and Application GUI

9.2 Demo Board Setup

Figure 20 (a) shows the board with the motor connected to **J8, J9, J10** and the power supply to **J3**. There are four push-buttons, a thumb-wheel potentiometer, a graphic LCD, and a few simple steps to quickly operate the motor out of the box. For debugging or programming, the user needs to connect **J5** with E1. Use the Mini-USB connector, **J1**, in the evaluation board for communication to the GUI.

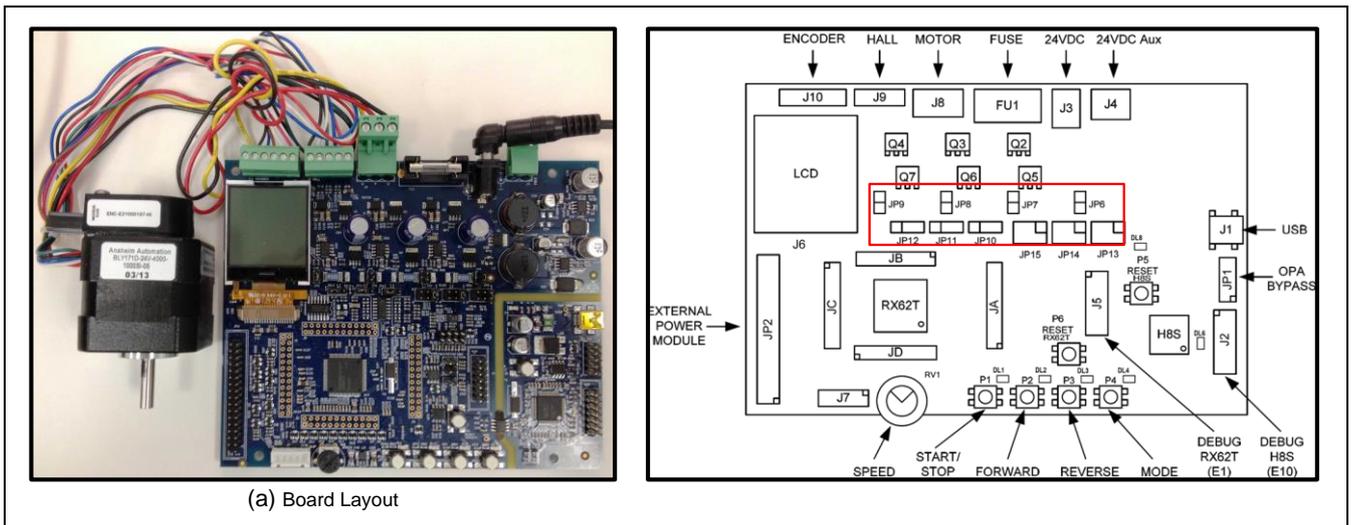


Figure 20 Demo Board Layout

Before starting the demo, reconfigure the jumpers (**JP6 – JP15**) as highlighted with red in Table 3. The jumper’s location is shown in Figure 20 (b). The board can be operated in standalone mode or in the GUI mode, but this demo algorithm only supports the GUI mode. The GUI mode will be explained in section 9.4.

Table 3 Configuration Jumper Settings

Operation		JP6	JP7	JP8	JP9	JP10	JP11	JP12	JP13	JP14	JP15
External OPA	3 - Shunt	1 - 2	–	–	1 - 2	1 - 2	1 - 2	1 - 2	1 - 2	1 - 2	1 - 2
	1 - Shunt	–	1 - 2	1 - 2	–						
Internal PGA	3 - Shunt	1 - 2	–	–	1 - 2	2 - 3	2 - 3	2 - 3	3 - 4	3 - 4	3 - 4
	1 - Shunt	–	1 - 2	1 - 2	–						

9.3 Build Project and Debug Operation with e²studio

To generate the firmware program for demonstration, the provided zip file must be imported to the project workspace using e²studio IDE revision 3.0 or higher. The following steps will explain the procedure for importing the project and setting up the debugger in the e²studio IDE.

9.3.1 Build Project Procedure in e²studio

Before importing the project, the user needs to install e²studio version 3.0 or higher and Renesas compiler CCRX revision 1.0. *Note: This demo will only use Renesas compiler CCRX revision v1.02.01.* The user will need to create a new file folder in Windows Explorer. Open the e²studio IDE as shown in Figure 21 and proceed with the following steps:

- Step 1.** Browse or type the newly created file folder path in the “Workspace Launcher” window and click the <OK> button.



Figure 21 e²studio IDE Start-up Windows and Workspace Launcher

- Step 2.** Select “Import” from the “File” pull-down menu.

After selecting “Import”, Figure 22 shows a *Select* dialog box that prompts the user to “Create new projects from an archive file or directory.”

- Step 3.** Select “Existing Projects into Workspace” from the *Select* pop-up dialog box and click the <Next> button.

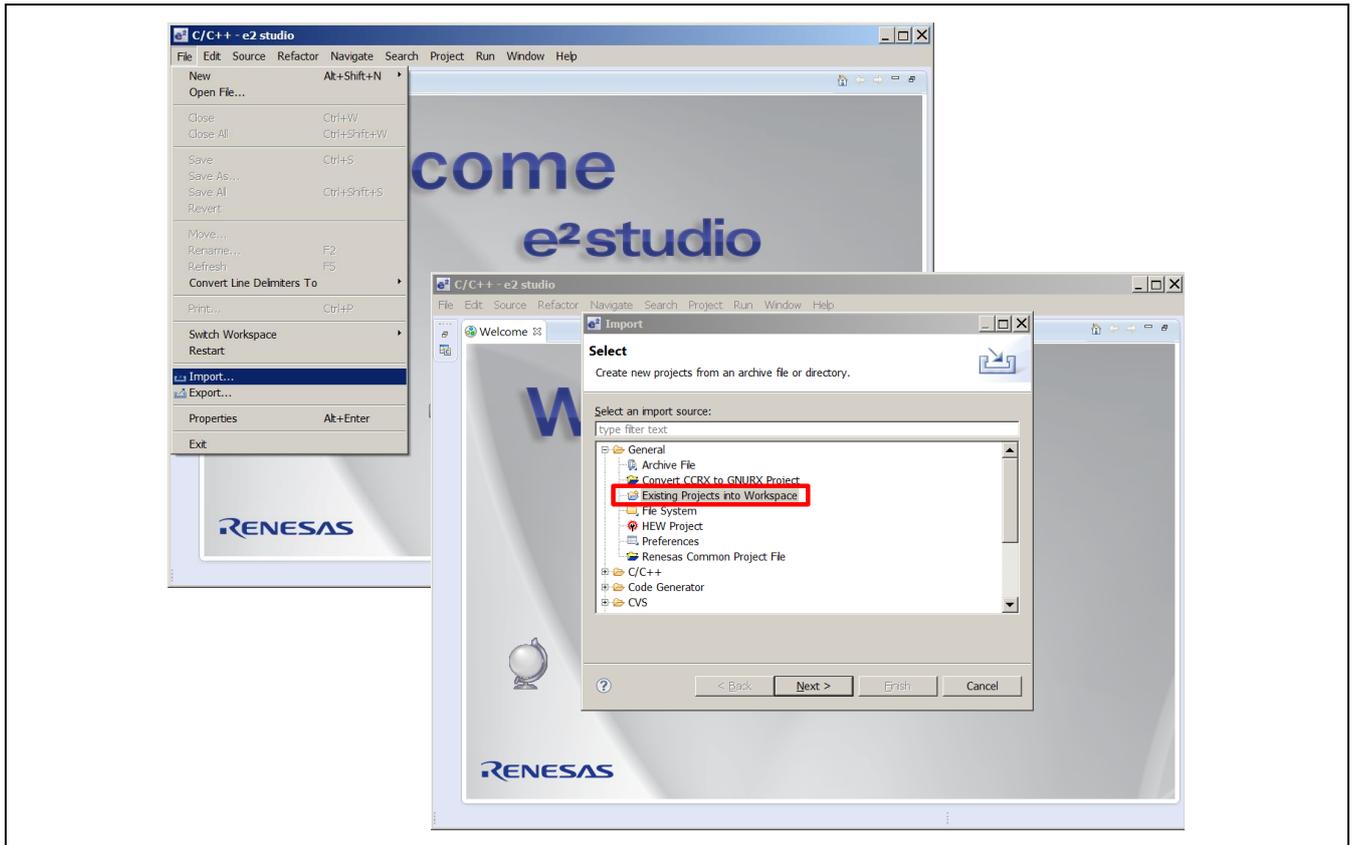


Figure 22 Importing Projects into the Workspace 1 of 2

After clicking the <Next> Button, the *Import Projects* dialog box in Figure 23 prompts the user to “Select a directory to search for existing Eclipse projects.”

- Step 4.** Select the Radio Button “Select archive file” and click <Browse> to locate the Position Control with Encoder zip file to import into the workspace.

The selected project will then appear with a checked box in the *Projects* message box as seen in Figure 23.

- Step 5.** Check the “Add project to working sets” check box, and then click the <Finish> button to complete the project import.

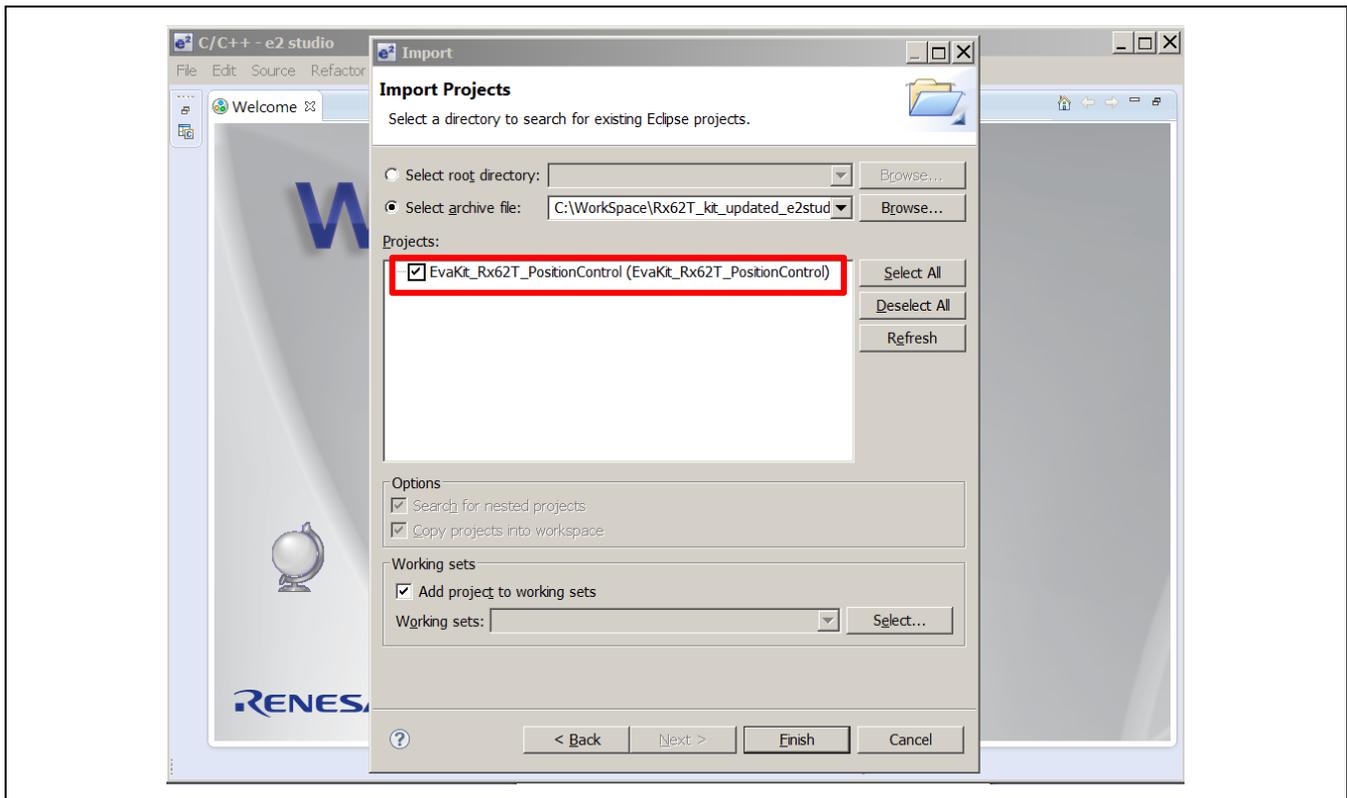


Figure 23 Importing Projects into the Workspace 2 of 2

If the file does not appear with a check box in the *Projects* message box, the selected zip file is the wrong zip file type, or it was not properly exported. If the file already exists in the workspace, then the user will see a message that states, “Some projects cannot be imported because they already exist in the workspace.”

After clicking <Finish>, the imported project is now in the e²studio workspace shown in Figure 24, and the project should be in “Debug” mode by default.

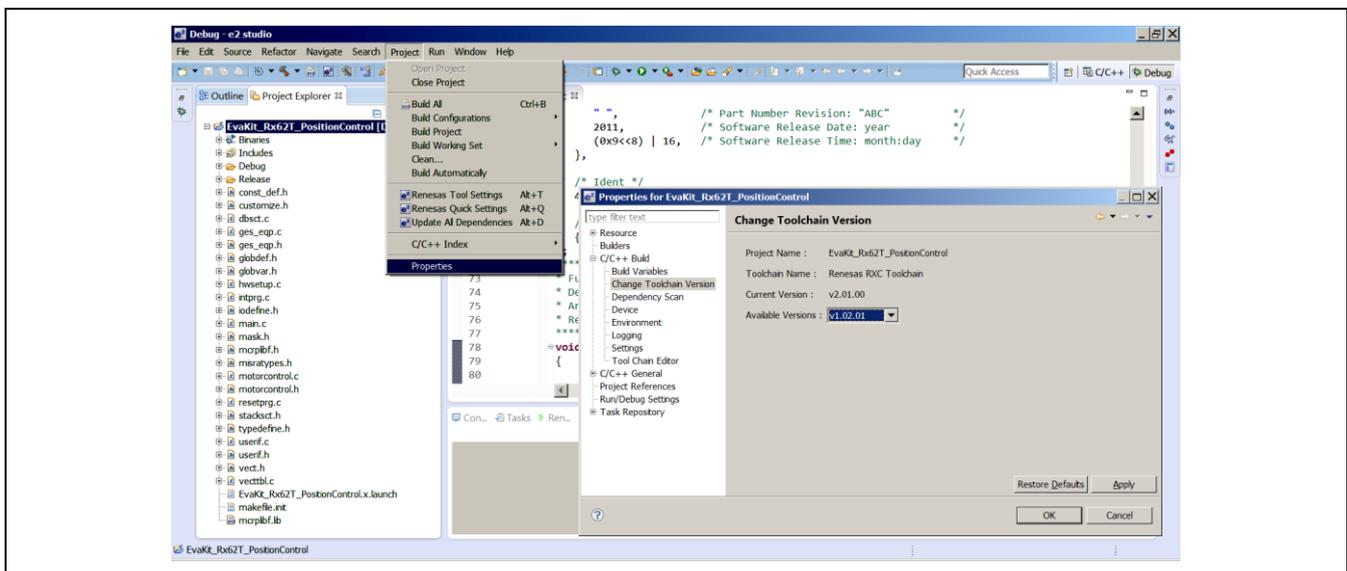


Figure 24 Setting the Toolchain Version in the e²studio Workspace

- Step 6.** Select “Properties” from the “Project” pull-down menu and expand “C/C++ Build” section. Select the “Change Toolchain Version” option and set the “Available Versions” to v1.02.01.
- Step 7.** Select the “Clean” command from the “Project” pull-down menu for cleaning and rebuilding the project.

Figure 25 shows the “Clean” Windows dialog box.

- Step 8.** Check the “Start a build immediately” option and select the Radio buttons “Clean all projects” and “Build the entire workspace.” Then click the <OK> button to clean and rebuild all projects in the workspace.

For debugging, the target firmware (.x file) is generated in the “Binaries” workspace folder shown in Figure 26. For release, set the active project to release mode for building projects. The target firmware (.mot file) is generated in the workspace “Release” folder.

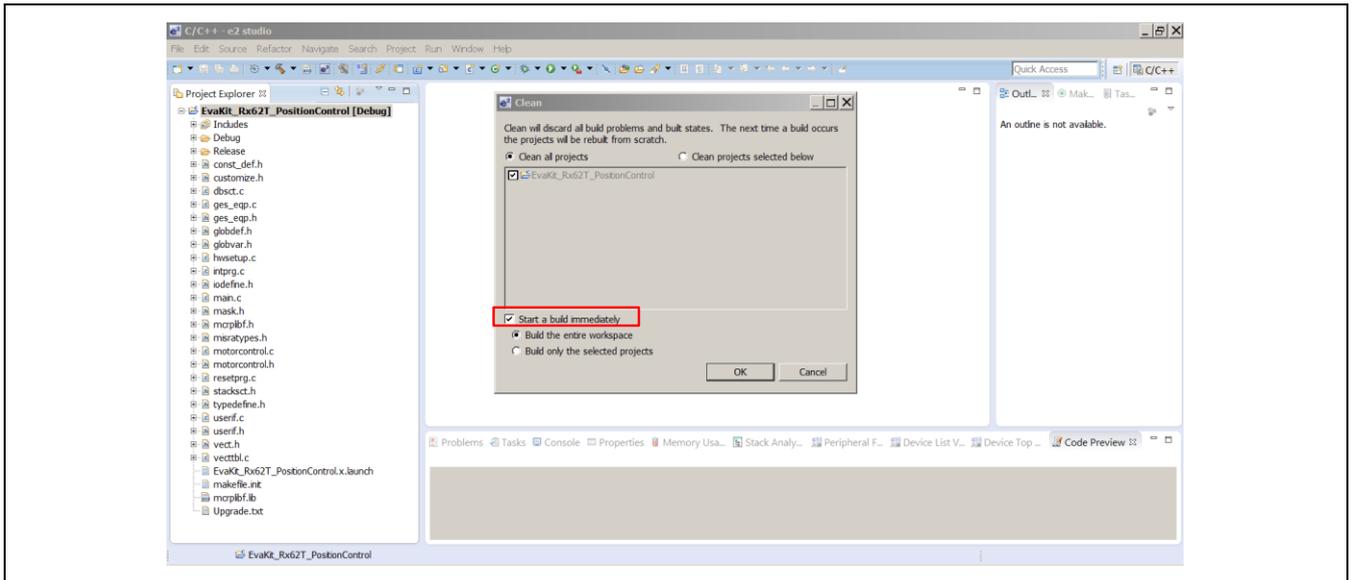


Figure 25 Clean Message Box

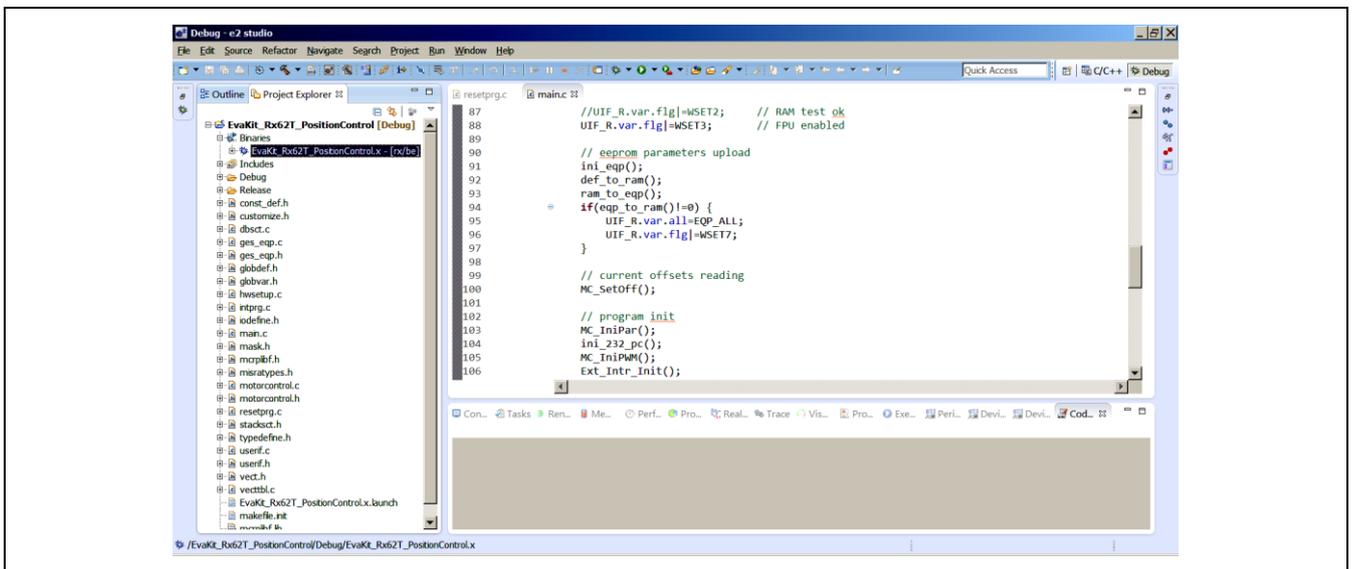


Figure 26 Target Firmware in Workspace

9.3.2 Debug Procedure in e²studio IDE

After generating target firmware, the user is now ready to setup the debug interface through the E1 debugger. The E1 debugger is necessary as an interface from the software to the hardware. Even if there is no need for any “debugging,” this procedure is still necessary to reprogram the board using the provided algorithms. Connect the 24V DC power to **J3**, the E1 Debugger to **J5**, and the motor to the **J8**, **J9**, and **J10** connectors. The connection is shown in Figure 27. Check the recommended jumper settings for this demo (refer to section 9.2).

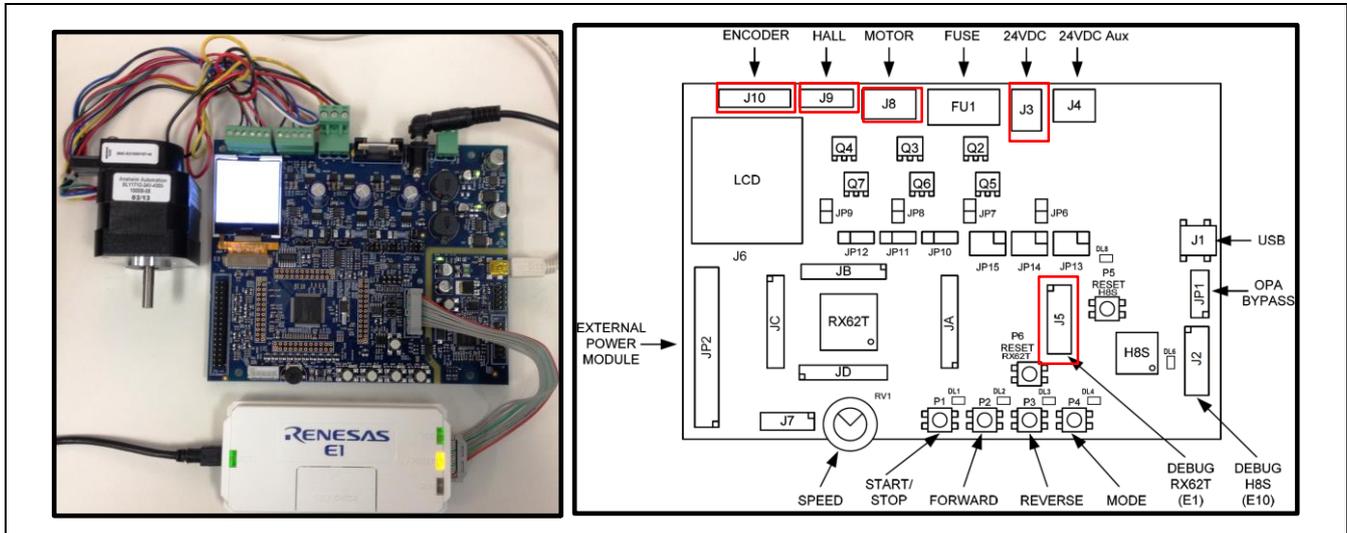


Figure 27 Debug Setup for the Demo Board

Step 1. Select “Debug Configurations” from the “Run” pull-down menu or click the debug drop-down icon [] and select “Debug Configurations”

Now, the user will view the “Debug Configurations” Windows dialog box, as shown in Figure 28.

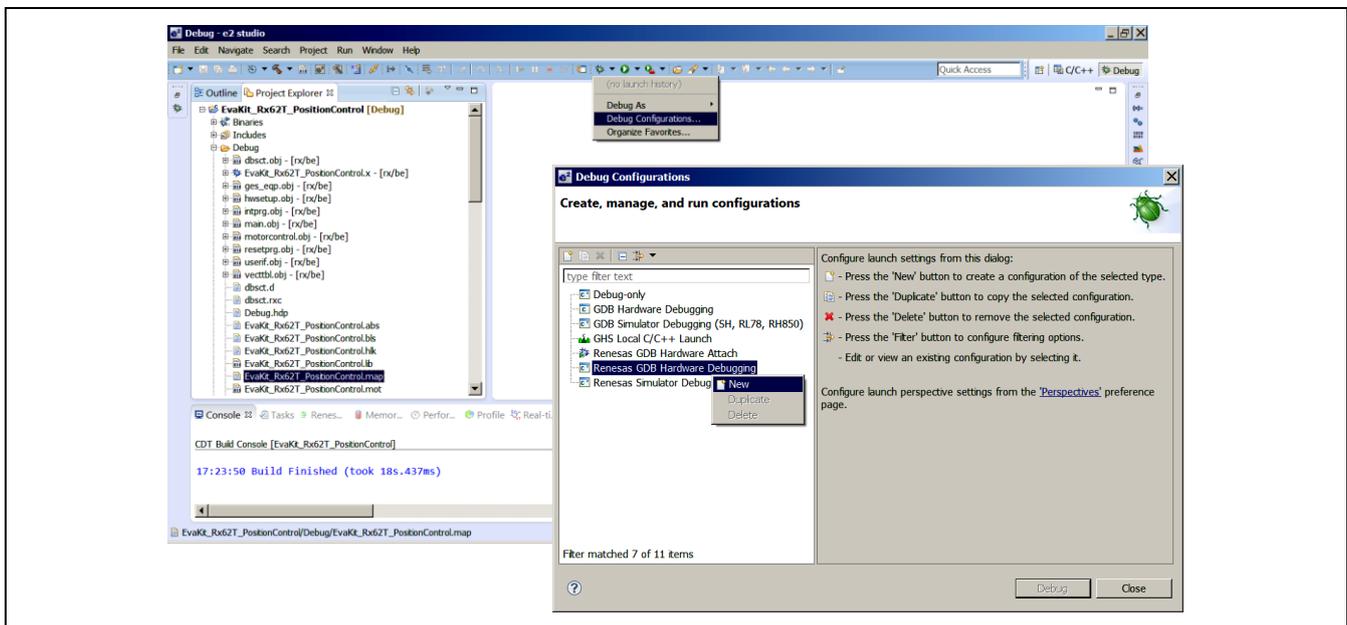


Figure 28 Setup Debug Configuration in Workspace

Step 2. Select “Renesas GDB Hardware Debugging”. Using the mouse, right click on “Renesas GDB Hardware Debugging” and select “New” as shown in Figure 28.

Step 3. In Figure 29 under the “Main” tab in Debug Configurations, Select the Position Control of PMSM with Encoder (EvaKit_Rx62T_PositionControl) as the “Project” and verify the “Build Configuration” tab is selected as “Debug” and the “Use workspace settings” is selected.

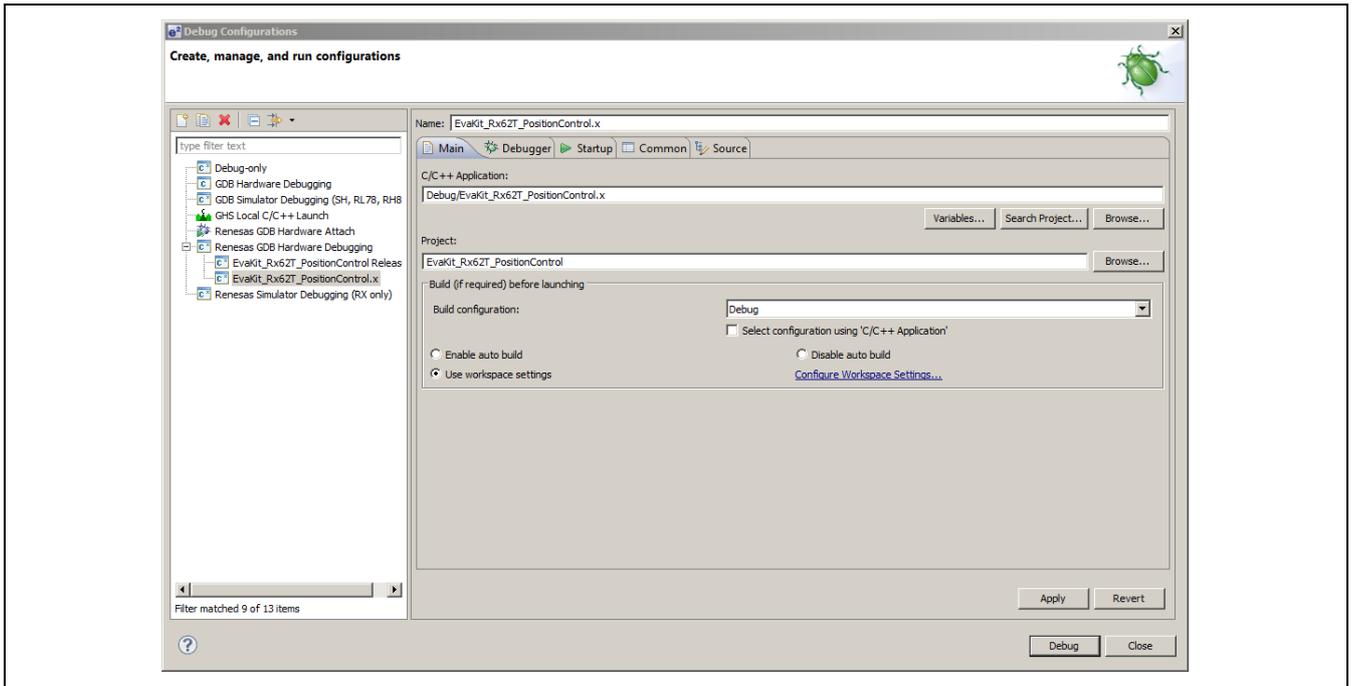


Figure 29 Debug Configuration Main Dialog Box

- Step 4.** Select the “Debugger” tab as shown in Figure 30.
- Step 5.** Select the “GDB Settings” sub-tab under the “Debugger” tab and set the “Debugger hardware:” to “E1” and “Target Device:” to “R5F562TA”.

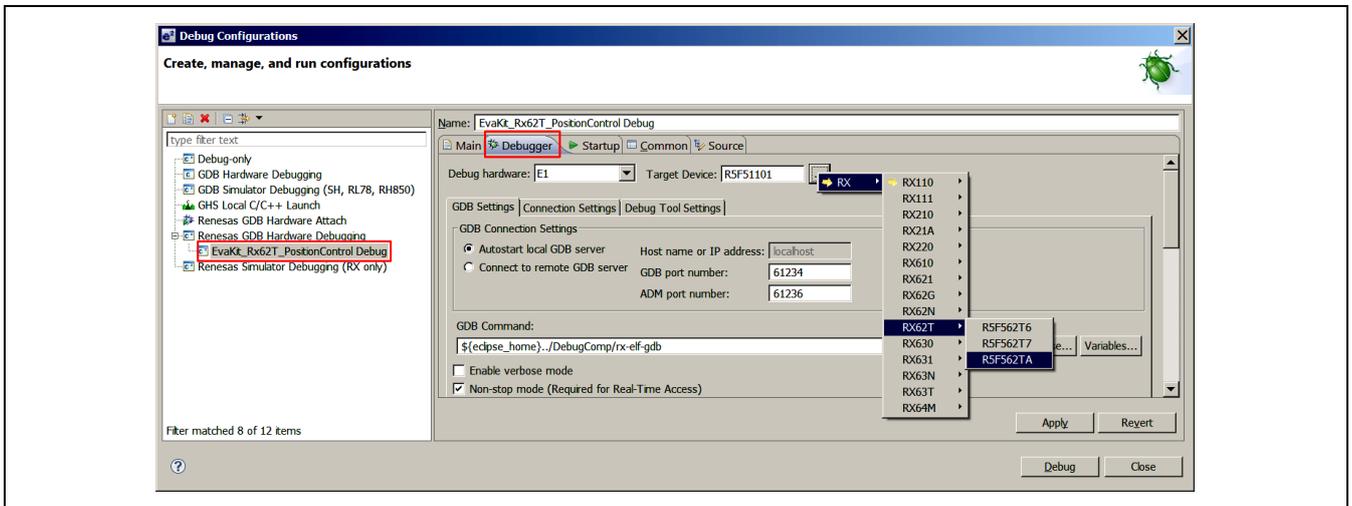


Figure 30 Debug Configuration Debugger Dialog Box

- Step 6.** Select the “Connection Settings” sub-tab and change the “External Frequency” value to “12.00 MHz” and “JTag Clock Frequency” to “12.38 MHz”. Set “Power” to “No.”
- Step 7.** Select the “Debug Tool Settings” sub-tab under the “Debugger” tab and select “Big Endian” in the “Endian” setting under “Memory.” Then click the <Apply> button.

The typical debug settings for this demo are shown with RED boxes in Figure 31 and in Figure 32.

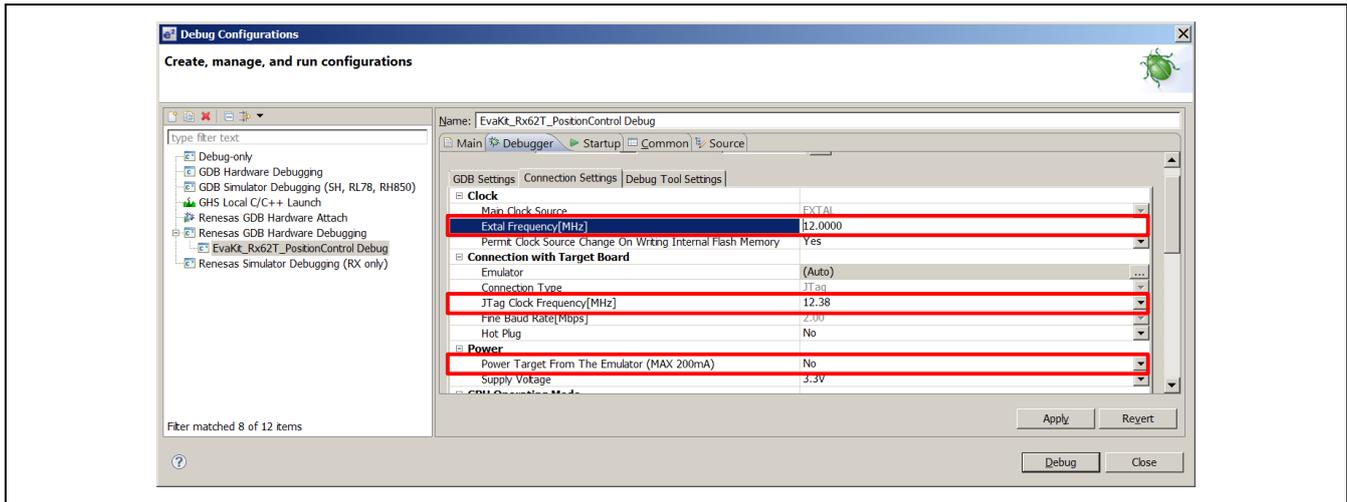


Figure 31 Debug Configuration Dialog Box 1 of 2

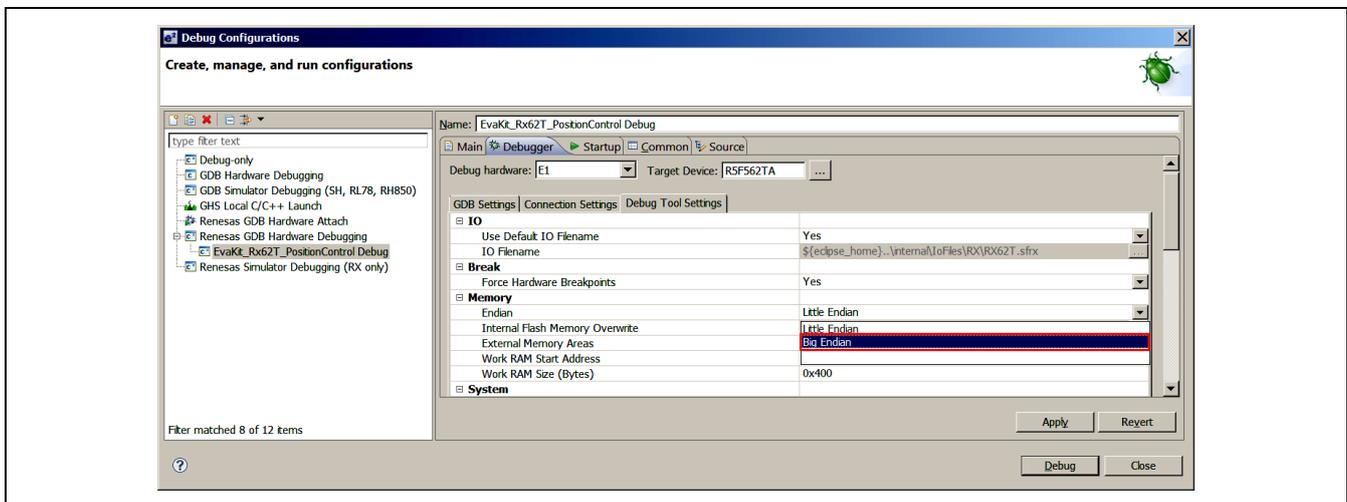


Figure 32 Debug Configuration Dialog Box 2 of 2

Step 8. Check the target board power is ON and verify the connections through the PC, E1 debugger and the target board.

Clicking the <Debug> button in the “Debug Configurations” dialog box will download the firmware to the target board.

Step 9. Click the “Resume” icon  or the F8 Key to run the program. This may require the user to press the “Resume” icon or F8 multiple times depending on the delay in the code. The icon should turn gray when the program is running.

The LED **DL1** will blink at about a one second rate continuously while running the target board.

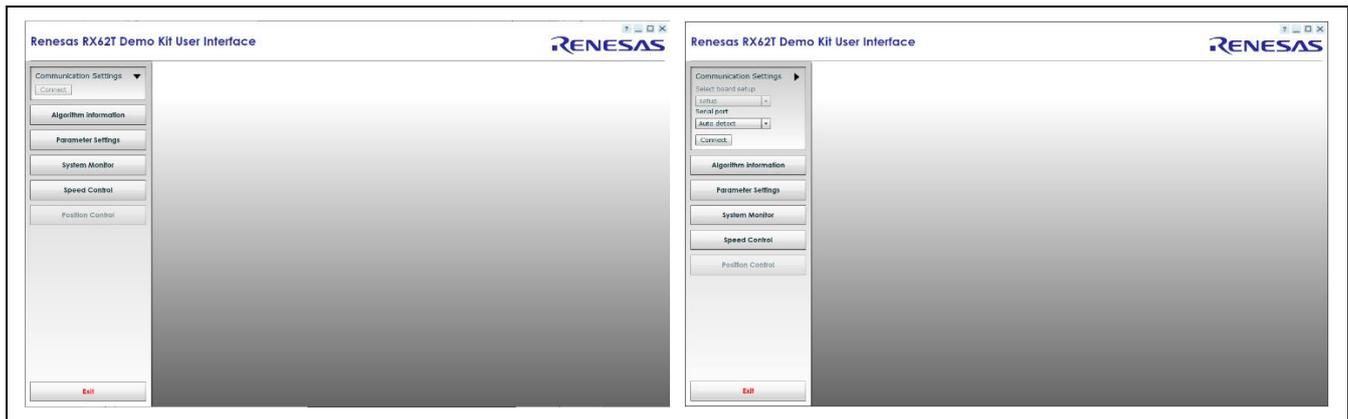


Figure 34 Connecting to the Motor Control Demo GUI 1 of 2

After successfully connecting with the target board, the “Communication Settings” area will change to a green color and the “Connect” button will change to “Disconnect.” The LED **DL6** will blink while communicating between the target board and GUI.

Figure 35 shows the GUI after a successful connection.



Figure 35 Connecting to the Motor Control Demo GUI 2 of 2

The GUI will detect the programmed algorithm. In this case the “Encoder Based Position Control” will be used. After connection, the “Position Control” button is active while the “Speed Control” button is grayed out. The user can check with the “Algorithm Information” message box which shows a valid algorithm. Clicking the “Verify Jumper Settings” button shows Table 3 in the GUI. Figure 36 shows the “Algorithm Information” dialog box. Follow the below procedure for using the GUI.

The LED **DL1** will be blinking continuously while running demo with no fault occurrence. If a fault occurs, the LED **DL2** will flash and **DL1** will remain illuminated without flashing. If a fault occurs, press **P6** (reset) and check if **DL1** begins to blink. If pressing **P6** does not fix the fault, disconnect and reconnect the E1 debugger and the Mini-USB and reprogram the board using the steps discussed in section 9.3.2.

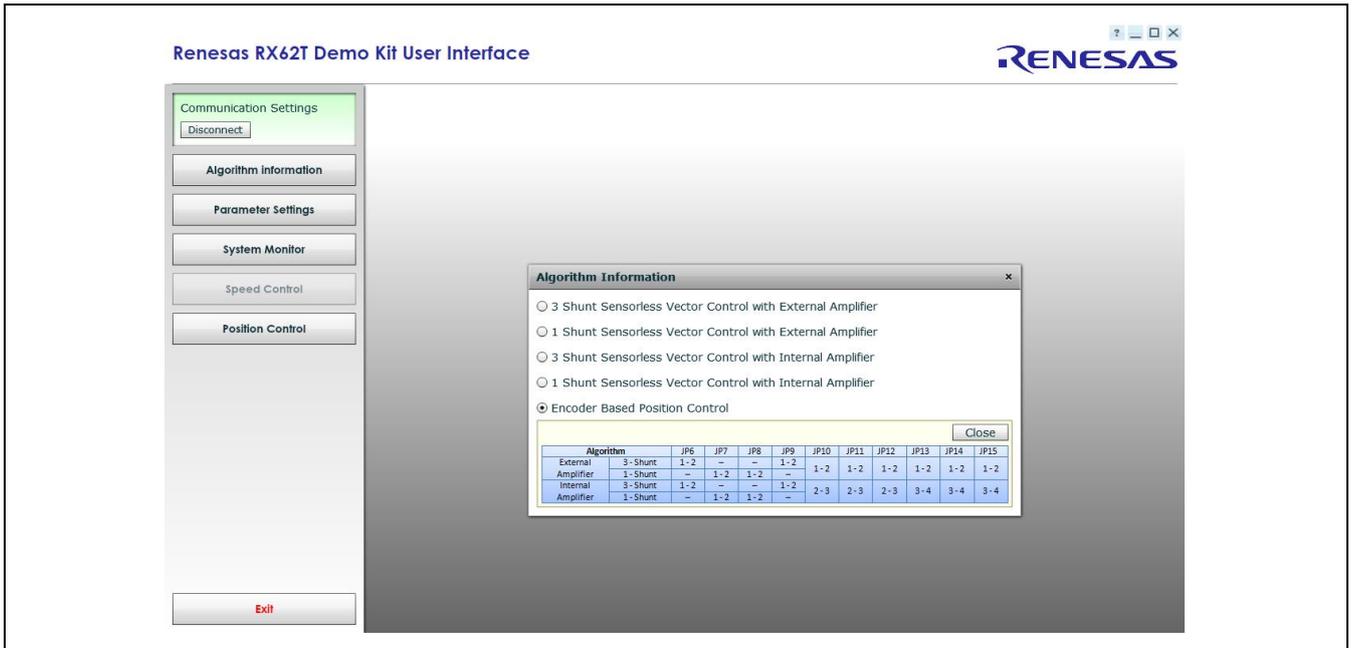


Figure 36 Algorithm Information in the GUI Application

Step 5. Click the “Position Control” button.

Step 6. Set the position arbitrarily by dragging the indicator needle to the right or left as shown in Figure 37, or a position can be manually typed into the dialog box below the indicator needle.

The motor shaft should rotate and stop at the set position degree. By default, the demo sets the below parameter values for position. Thus user needs to set angular position value from 5 to 8000 degrees to run this demo.

- Minimum position 5 degree
- Maximum position 8000 degree

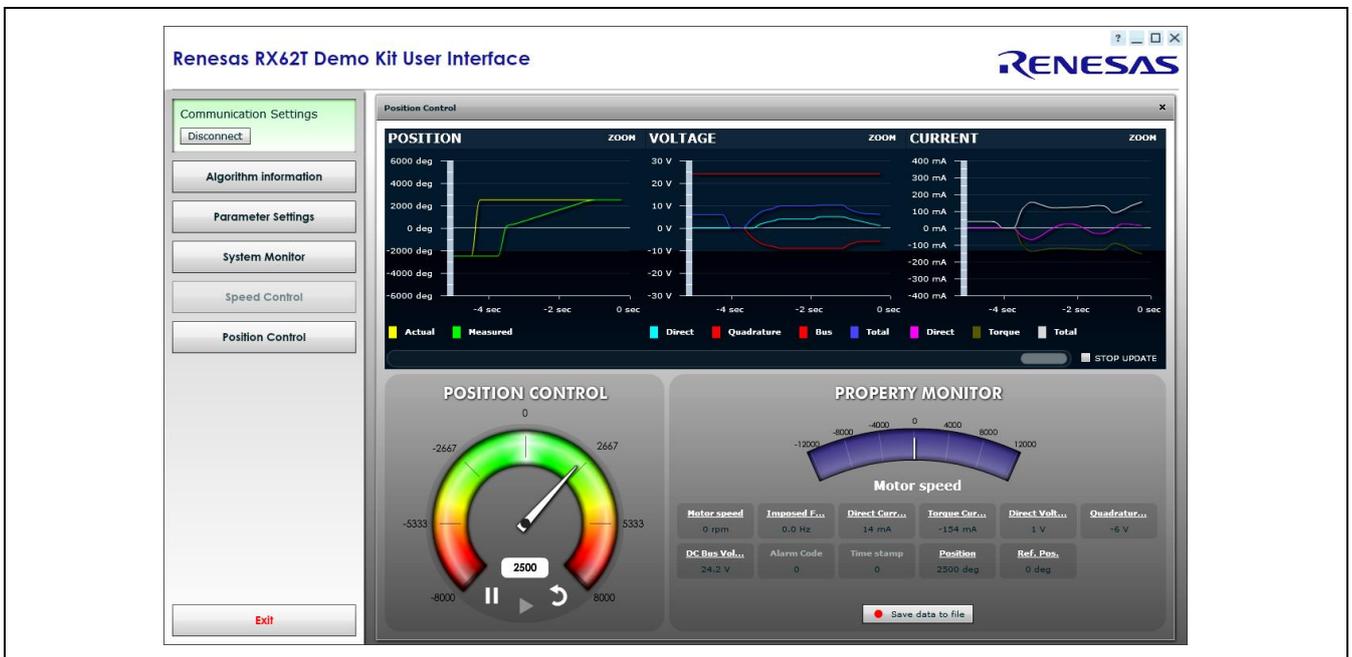


Figure 37 Set Position in Degree to Turn the Motor in the GUI Application

Step 7. Click the “Parameter Settings” button

The “Parameter Settings” feature can be used to manually adjust the preset variables using the GUI. Figure 38 shows the “Parameter Settings” dialog box.

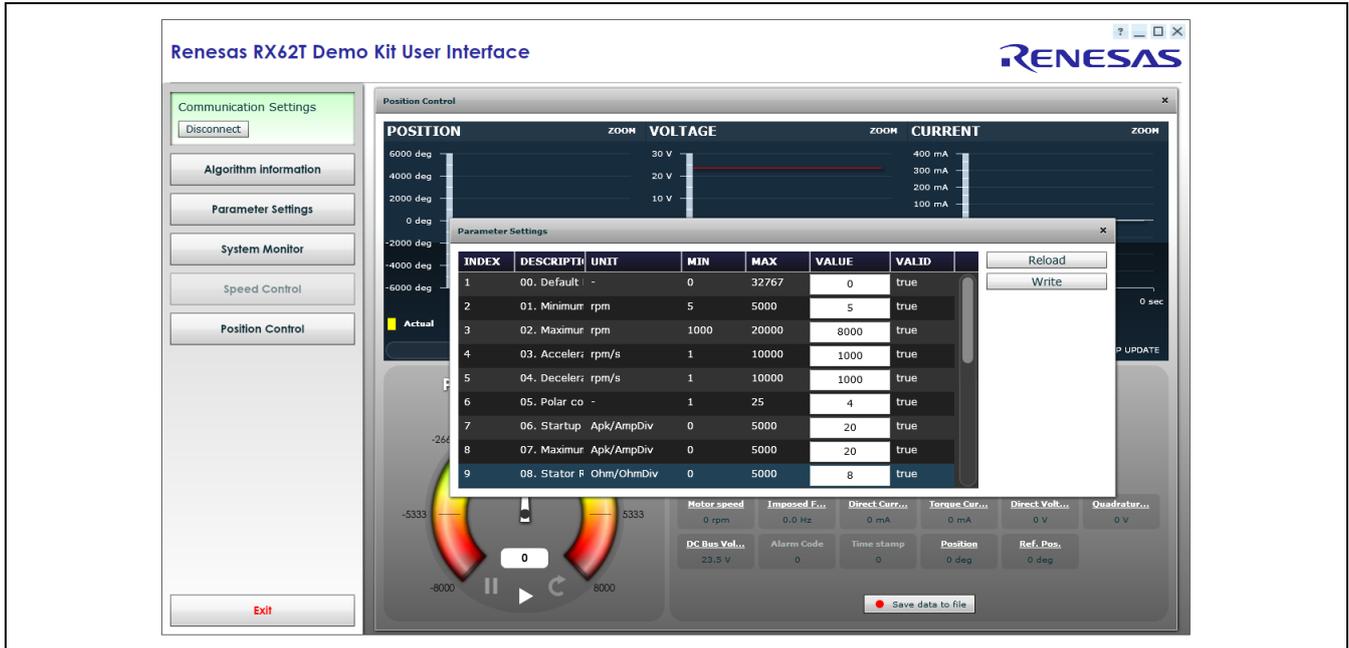


Figure 38 Changing the Parameter Settings in the GUI Application

To terminate the GUI application, return the control needle to zero position, press “Disconnect” button and then press the “Exit” button to close the application.

This concludes the description of the Position Control of PMSM with Encoder Demonstration Guide.

Appendix A - References

1. RX62T Group User's Manual: Hardware, R01UH0034EJ0110, April 20, 2011
2. DevCon 2010 Courses:
 - ID-620C, Complete Motor Control Integration with RX62T.
 - ID 623C, Understanding Sensorless Vector Control with Floating Point Unit (FPU) Implementation.
3. DevCon 2008 Courses:
 - ID-504, Speed Control Using a Digital Encoder and Vector Formulation
4. Application Note of Sensorless Vector Control of Three-Phase PMSM Motors, REU05B0103-0100/Rev.1.00, March, 2009
5. Application Note of Mcrp05: Brushless AC Motor Reference Platform, REU05B0051-0100, Feb, 2009

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 18, 2011.	—	First edition issued
2.00	Jan. 31, 2014.	—	Second edition issued
2.01	Jul. 30, 2014.	21	Demonstration Guide added

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under “General Precautions in the Handling of MPU/MCU Products” and the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

Handle unused pins in accordance with the directions given under “Handling of Unused Pins” in the manual.

- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

- The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses. The correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences Between Products

Before changing from one product to another (i.e. to a product with a different part number) confirm that the change will not lead to problems.

- The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
 2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
 3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
 4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
 6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
 7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
 11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

Refer to "<http://www.renesas.com/>" for the latest and detailed information.

Renesas Electronics America Inc.

2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH

Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.

7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.

Unit 301, Tower A, Central Towers, 555 LanGao Rd., Putuo District, Shanghai, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited

Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.

13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.

80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.

Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.

12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141