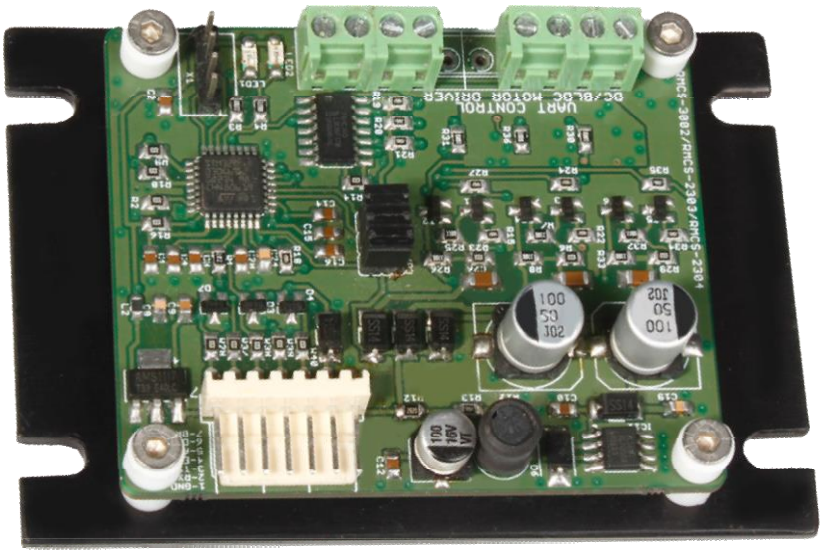
DC Servo Drive

10V - 30V DC, 50W with ASCII Modbus

RMCS – 2303



Operating Manual v1.0

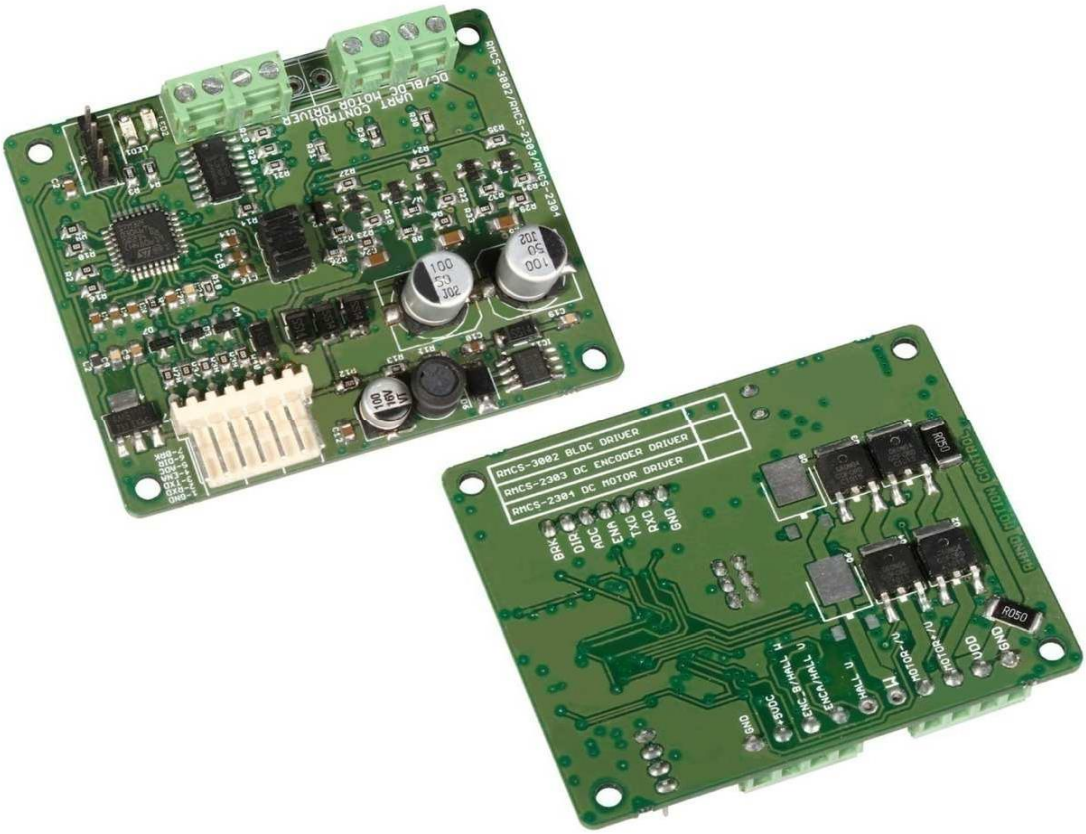
Contents

Introduction – Salient Features

Rhino Motion Controls RMCS-2303 with UART ASCII is a high performance dc servo drive (10–30 V DC) designed for optimized operation of any DC Servo motors with quad encoder feedback. This is an amazing cost effective solution to provide closed loop servo control for various applications. The salient features of this drive:

- This drive provides a closed loop speed and position control for DC motor with encoder.
- The motor programmed speeds are maintained irrespective of the voltages supplied.
- Also by using this drive, the rated torque of the motor is available at all speeds and the torque does not decrease with change in speeds.
- It is possible to run the motor in three different modes, Analog speed control mode, Digital speed control mode and Position control mode.
- It has short-circuit protection for the motor outputs, over-voltage and under-voltage protection and will survive accidental motor disconnects while powered-up.
- This drive is configured using MODBUS ASCII protocol via UART.
- There is a function in the drive for setting the Modbus Slave Address from 1 to 7 using physical jumpers (Hardware Setting) or using PC GUI or any other device like PLC or microcontroller which can provide proper data output on ASCII Modbus.

- There are three user modes in the drive :
  - Mode 0 - Analog Control Mode
  - Mode 1 - Digital Speed Control Mode
  - Mode 2 - Position Control Mode

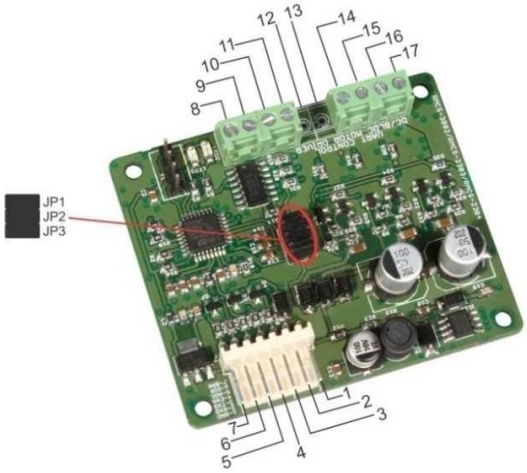


Technical specifications and Pin description

Supply Voltage and Current

Specification	Min	Max	Units	Comments
Supply Voltage	10	30	Volts DC	Between +Ve and GND
Phase Current	0.5	3	Amps	Peak 3 Amps per phase

Pin description of the drive is as per below image:



Pin No.	Description
1	GND
2	RXD
3	TXD
4	ENABLE
5	ADC
6	DIRECTION
7	BRAKE

(Pins 1-7 are used for drive Configuration and UART control)

Pin No.	Description
JP1	Jumper 1
JP2	Jumper 2
JP3	Jumper 3

(JP1 to JP3 are used for Hardware setting of slave ID)

Pin No.	Description
8	GND
9	+5VDC
10	ENC_B/Hall W
11	ENC_A/Hall V
12	Hall U
13	W
14	Motor- / V
15	Motor+ / U
16	VDD
17	GND

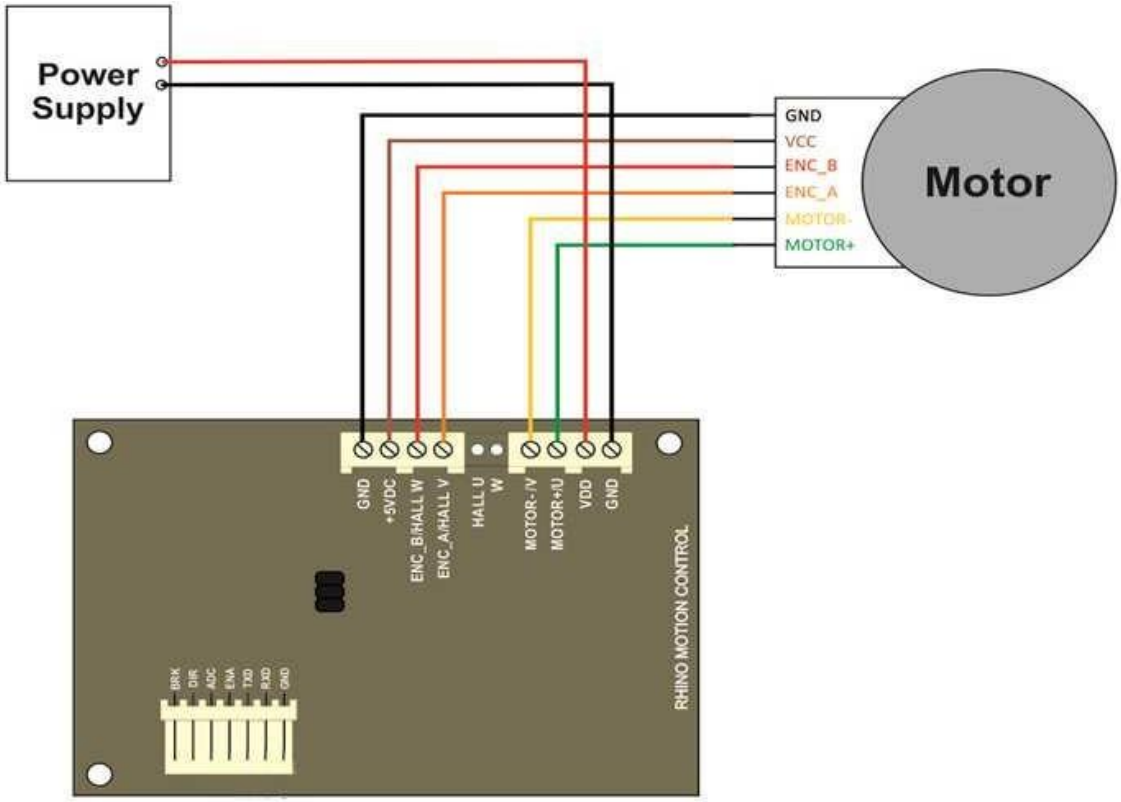
(Pins 8 - 17 are connected to motor and power supply as described)

Drive & Motor Connection

Drive Pin outs	
Pin No.	**Description**
8	GND
9	+5VDC
10	ENC_B/Hall W
11	ENC_A/Hall U
12	Hall U
13	W
14	Motor- / V
15	Motor+ / U
16	VDD
17	GND

Motor Pin outs	
Motor	**Wire Color**
GND	Black
VCC(5 V DC)	Brown
ENC_B(Encoder B)	Red
ENC_A(Encoder A)	Orange
M-(Motor-)	Yellow
M+(Motor+)	Green

Power Supply Pin outs
VCC – 10 to 30V
GND



Modes of Operation

The drive can be configured in the following three modes

1. **Mode 0: Analog speed control mode:**

    a. In this mode the speed of the Rhino DC Servo motor can be controlled by an externally connected Potentiometer.
    b. Speed can be increased or decreased manually based on requirement using potentiometer.
    c. The drive will provide full torque at all speeds within the range.
    d. However the potentiometer has to be connected via a voltage divider to provide a maximum of 3.3 volts so as to not damage the drive.
    e. Also the Enable, Brake and direction pins have to be connected as per configuration requirements.

2. **Mode 1: Digital Speed control mode with direction control:**

    a. In this mode the speed and direction of the DC servo motor is settable / controllable via a Computer or any microcontroller board like Arduino or PLC or any other Modbus ASCII compatible device.
    b. Like the analog mode here there is no compromise in the torque output of the motor irrespective of the operational speed and voltage supply and control at higher speeds.
    c. This mode can be used when multiple motors are to be used to run at exactly the same RPM and same torque even though the voltage supply might be different.
    d. Also in this mode the direction of the motor can be controlled digitally via Modbus ASCII commands to run the dc servo motor in both directions
    e. In applications like conveyor belts where speed control is critical and industrial robotic applications like solar cleaning robots where straight line motion is critical for correction operation of the equipment, the digital speed control mode can provide optimal results.

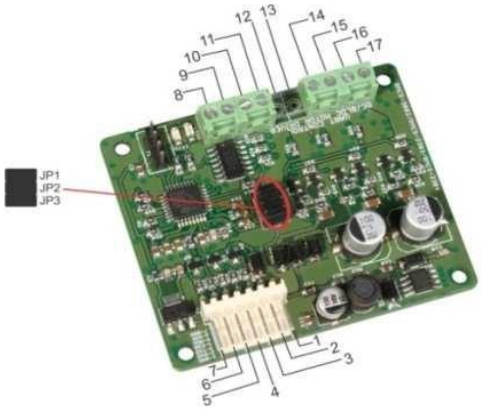
3. **Mode 2: Position Control Mode:**

    a. In this mode Speed, Direction, Acceleration and Position of motor can be controlled.
    b. The position control mode is suitable in applications where exact movement of motor is required. This can be used in precision applications like machine control, motion control or robotics.

In all modes absolute encoder position and speed is available on Modbus.

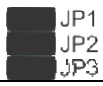
Slave ID Addressing

Multiple drives with different slave addresses can be used with a single controller on single UART bus. For this all drives should have a unique Slave Address. Using the GUI software or Modbus commands, the slave ID can be set from 1 to 247. However, slave ID from 1 to 7 can be set using physical jumpers also. As shown in the image below there are three jumpers marked by a red circle shown in the image below.



The three jumpers JP1, JP2 and JP3 can be set in the configuration as per the below table to provide a physical slave address to the drive. In the below table a value of '0' corresponds to a state where the jumper is not connected and a value of '1' corresponds to a state where the jumper is connected. If none of the jumpers are connected the drive has been programmed to use the Slave ID 11 in default mode which can be changed from 1 to 247 using GUI software or Modbus commands.

Drive will check jumpers on startup. If no physical jumper is connected it will use programmed slave id.

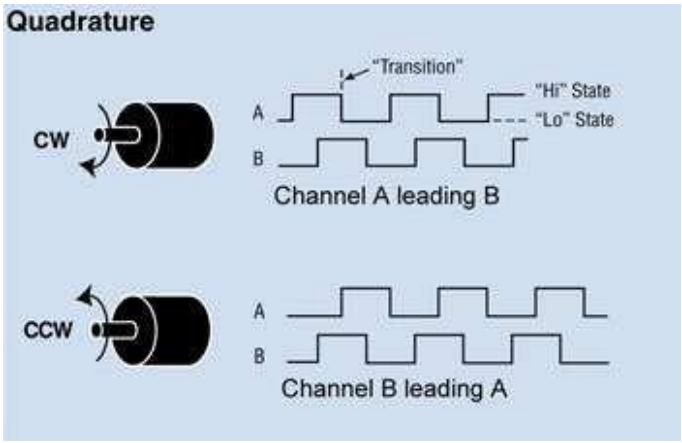
Slave ID	JP1	JP2	JP3	Image of connection on the Drive
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

Basics of a servo drive

This is a servo drive to work with quad incremental encoder and dc motor. When the drive starts up the encoder position is set as 0. With reference to the start position drive will always have a 32 bit number keeping track of movement. It can be set to 0 at any point.

This servo drive works with quadrature encoder. The quadrature encoder also known as quad encoder or incremental encoder gives two channels of pulses output. It can be of optical or magnetic type.

The basic concept is the encoder gives output of two channels of pulse streams say channel A and channel B to drive, when A is leading B or B is leading A the drive can also determine the direction of movement.



So drive can calculate the speed and direction of motor according to the pulses if number of Lines per Rotation is known. For each line drive gets 4 counts, for example if encoder is 334 lines drive can get 1336 (334 x 4) counts per rotation. This parameter can be set in register 10. It's a 16 bit unsigned number having range of 1 to 65535.

In any of the 3 modes drive manages speed, direction, acceleration, deceleration, position of motor and voltage and current given to the motor based on set parameters, gains and feedback from encoder. This is a close loop system means in position mode the drive always tries to make the error which is difference between the required position and error to 0, in case of speed mode it tries to make the speed same as set speed.

The voltage and current given to motor is also managed by drive to provide higher torque when required. This also means that even at lower speed, higher torque up to the maximum torque available in motor is possible.

How to use the drive:

The drive needs to be configured to run in any one of the three modes described in the above section to make a closed loop control system. The drive can be configured using a PC with GUI software / Modbus controller / Arduino Board.

To configure the drive for a closed loop system

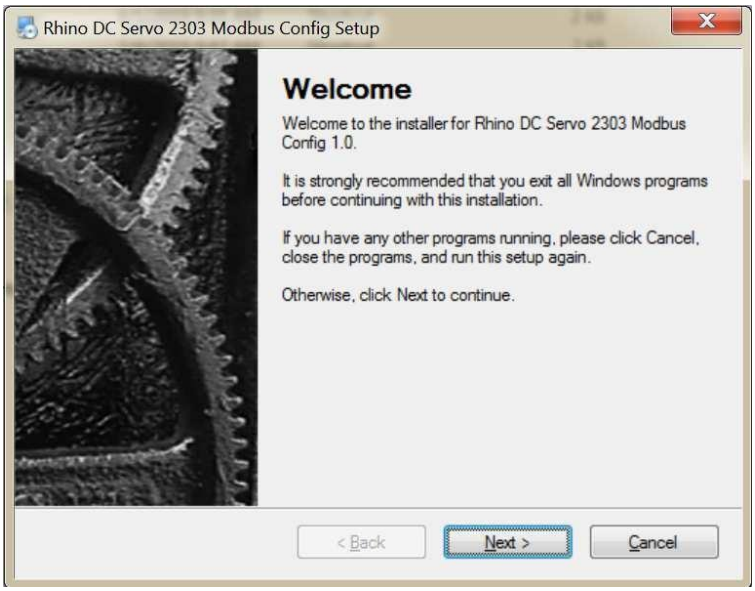
- PC with GUI software for Rhino 2303 drive or any generic Modbus software (Like Modbus Poll)
  - GUI software https://robokits.co.in/downloads/Rhino%20DC%20Servo%202303%20Config%20Setup.exe
  - Modbus Poll demo version - https://www.modbustools.com/download.html
- RMCS-2303 UART ASCII Encoder DC Motor Driver
  - https://robokits.co.in/motor-drives-drivers/encoder-dc-servo/rhino-industrial-encoder-dc-motor- driver-50w-with-uart-ascii-compatible-10-to-30-v-10a
- Encoder DC Servo Motor (Any DC motor 10-30VDC 50W max with encoder)
  - https://robokits.co.in/rhino-planetary-encoder-dc-servo
  - https://robokits.co.in/rhino-ig32-precision-dc-servo
  - https://robokits.co.in/motors/encoder-dc-servo/high-torque-dc-encoder-motor
  - https://robokits.co.in/motors/encoder-dc-servo/high-torque-high-precision-motor
- Industrial Power Supply (below is a recommended supply. It can vary as per your requirements)
  - https://robokits.co.in/power-supply/industrial-power-supply/24v-10a-industrial-power-supply
- USB UART Module
  - RKI-1154 CP2102 - https://robokits.co.in/control-boards/interface-boards/cp2102-usb-uart-module
- External 10K Potentiometer and resistors (for analog Speed Control mode only).

Software Installation:

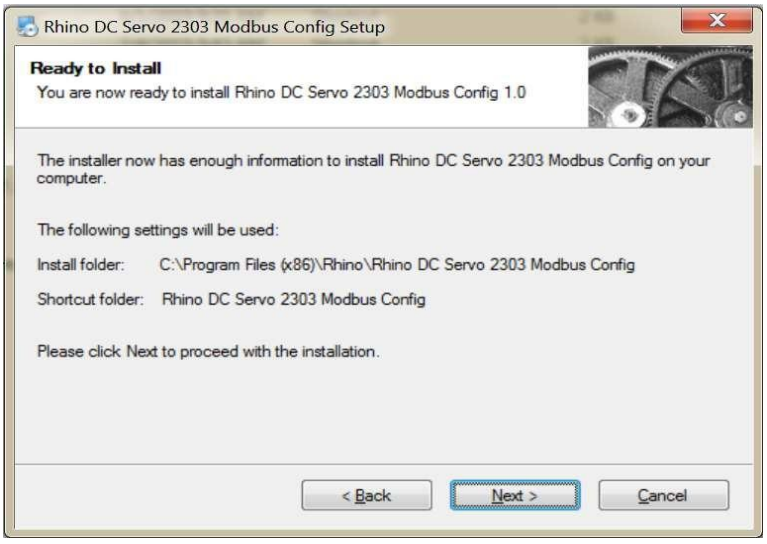
Software is available here
https://robokits.co.in/downloads/Rhino%20DC%20Servo%202303%20Config%20Setup.exe

Download and run the setup executable.

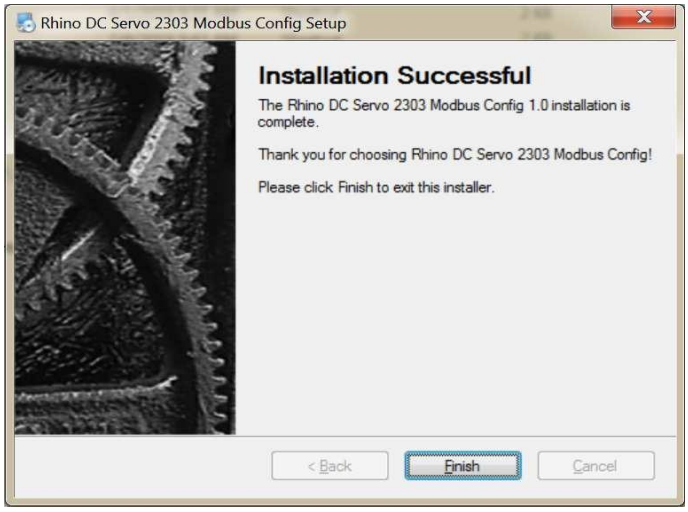
- Click **Next** button



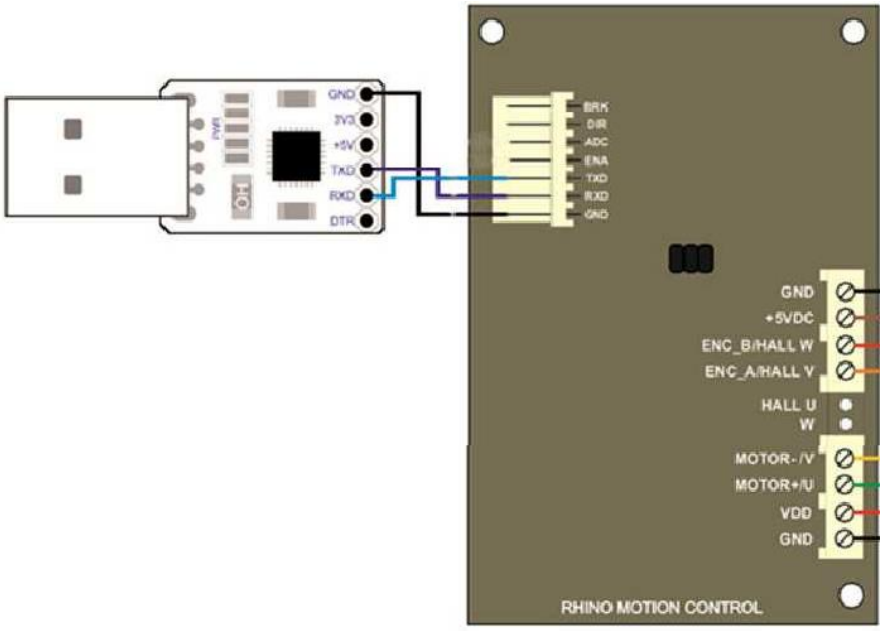
- Verify installation destination and click **Next**



- Click Finish when Installation Successful message shows up.

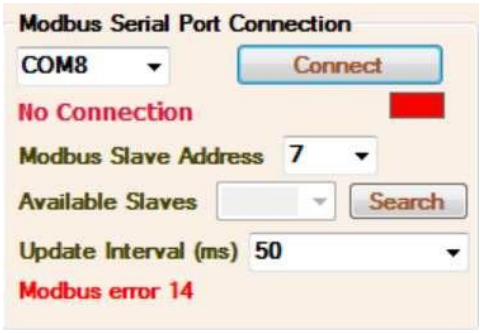


Now, Software is ready to use. find the shortcut on desktop and open the software. Once the Software is installed connect USB-UART as per shown in below figure.

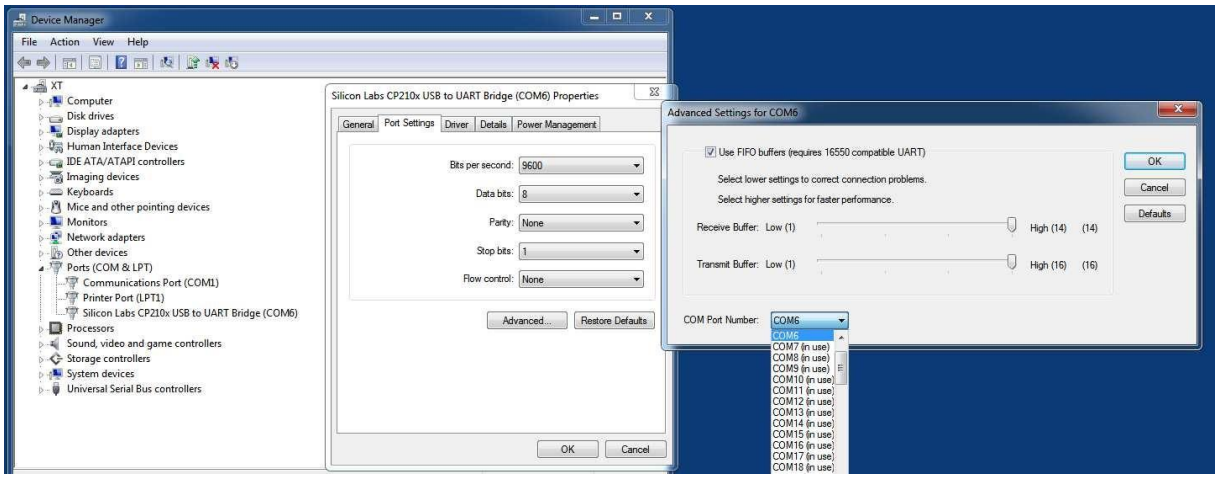


COM port Selection

Select COM Port for USB-UART, then click on "Connect" button.



You may look in 'Devices and Printers' or 'Device Manager' for specific serial port of your device. The software supports serial ports up to COM32 only. If your device's port number is higher you can change it in - Device Manager > Ports (COM & LPT) > Double click on device name > Port Settings > Advanced > COM Port number



Once your drive is connected to COM Port and it shows "Connected" message.

When the drive communicates with software You can see a blinking Green indication.

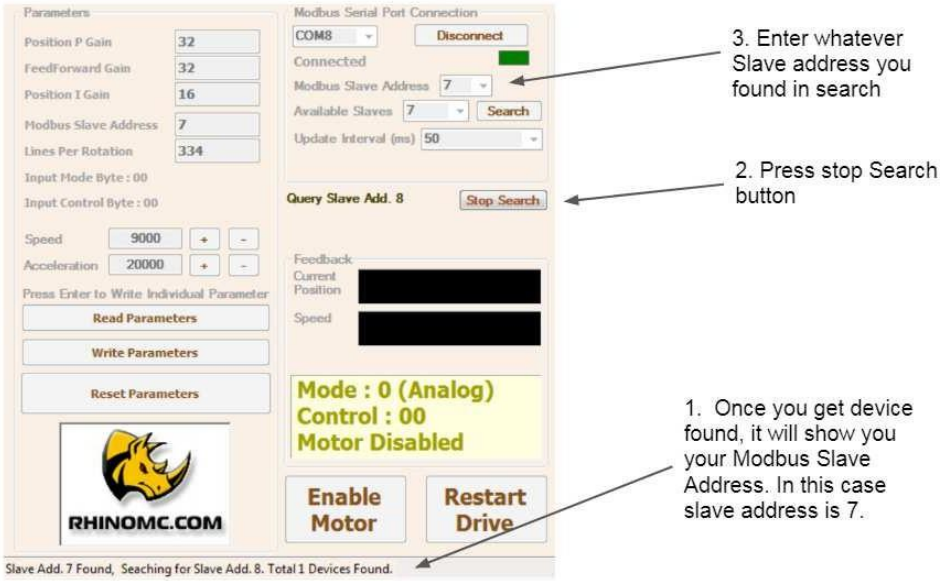
Selection of Modbus Slave Address:

There are two ways to select Modbus Slave address.

1. If Slave address is already known, it can be directly selected from drop box.

2. If slave address is not known or multiple drives with multiple addresses are connected on serial bus, click search button. This will search for all slave ids from 1 to 247. If any response is received from slave id it will give indication in status bar and list the drive in 'Available Slaves' drop down box.

To search slave address click on "Search". Once all the connected devices are found click on "Stop Search". Select any of detected slave address in Modbus Slave Address drop down box and drive will start communicating to software.



Make sure no multiple drives with same slave address is connected on serial bus.

Once any drive is connected and slave id for the same is selected, software will read parameters of the drive. It can also be done manually by clicking 'Read Parameters' button.

Parameters

READ PARAMETERS

"Read Parameters" is used to read current parameters of drive.

SET A PARAMETER

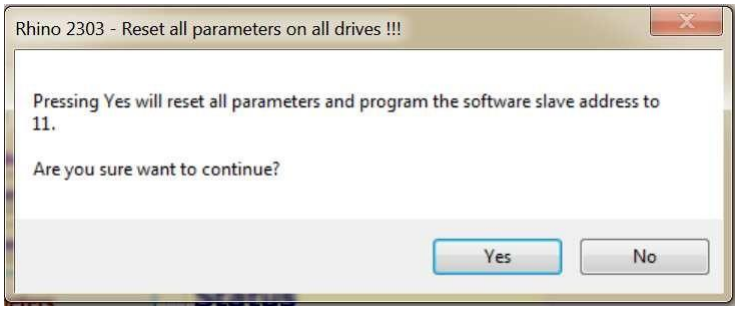
- Pressing 'Enter' on text box will change the parameter in drive but it will not be saved permanently. 'Write Parameters' button must be clicked to save all parameters permanently.
- Set Position Proportional gain, Velocity Feed Forward gain and Position Integral Gain as per requirement. These can be used to remove vibrations and making movement of motor accurate.
- Set Modbus slave address and lines per rotation. Lines Per Rotation is very important because it does have effect on indication of speed feedback.
- If slave address is changed the software will automatically change the slave id of currently connected device.
- Set Speed and Acceleration as per the requirement of application.

WRITE PARAMETERS

Once all the parameters are entered and tested, click on "Write Parameters" to save all parameters in drive. These parameters will be changed permanently. Notification will be shown as below when all parameters are written successfully.

RESET PARAMETERS

To reset the drive to default configuration click 'Reset Parameters' button. This will set all parameters to default values. When button is clicked software will ask for confirmation.



Click on "Yes" to reset the drive. After Clicking on 'Yes' if reset is successful confirmation message will be shown. Power off and Power on the drive to load default parameters. If any physical jumpers are preset at this point the slave id of the drive will be according to that otherwise slave id will be 11.

Control Modes

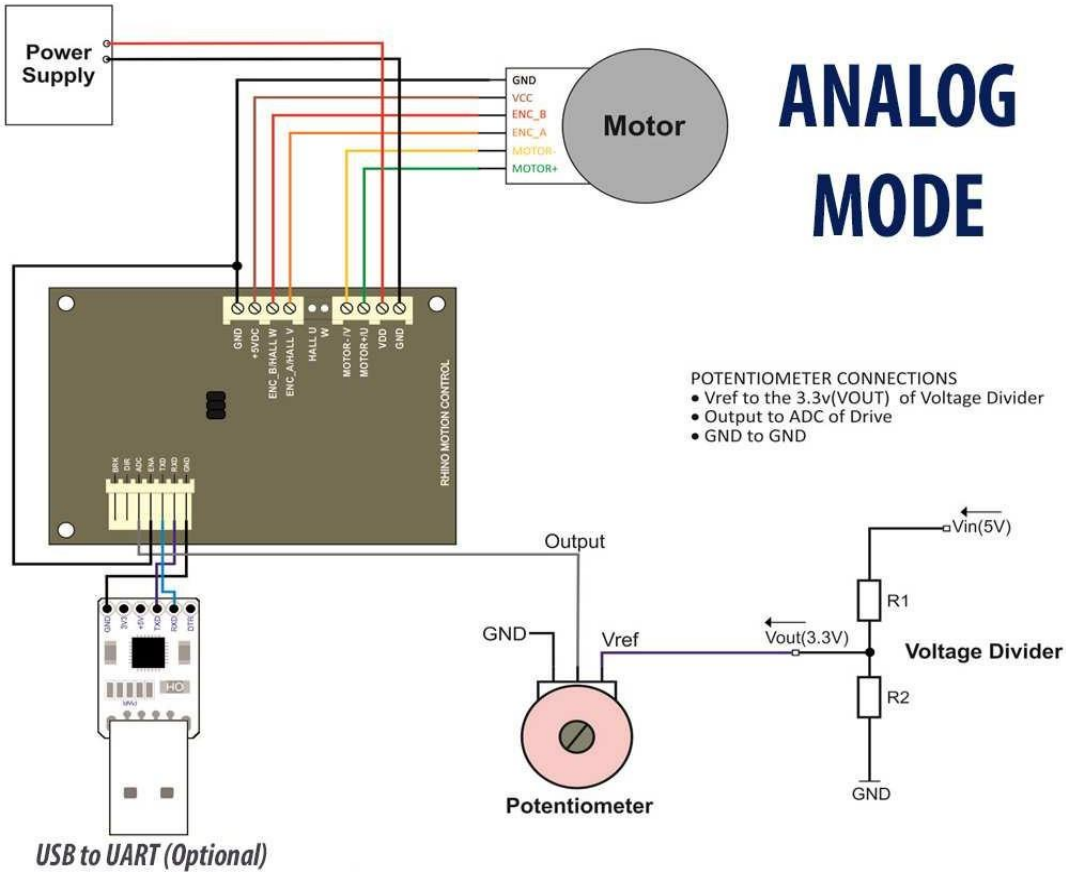
Motor can be run in 3 different modes - 0: Analog Mode, 1: Digital Speed Mode, 2: Position Mode

Mode 0 - Analog Control Mode

In this mode drive can be controlled using an external potentiometer or analog voltage input. As this is a close loop drive, high torque (up to the maximum torque motor can provide) is available even at low speed.

- Connect external potentiometer to change speed of motor as per diagram.
- Connect enable (Pin 4) of drive to GND (Pin 8) to enable motor in analog mode.
- Direction of motor can be changed in analog mode by connecting the Direction (Pin 6) to the GND (Pin 8).
- Electronic brake can be applied to motor by connecting Brake to GND - (Pin 7) to GND (Pin 1).

Hardware Connection:



Drive to PC/Analog Connection

Drive Pin outs	
Pin No.	Description
1	GND
2	RXD
3	TXD
4	ENA
5	ADC
6	DIR
7	BRK

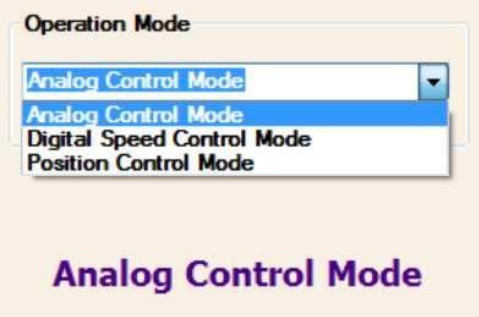
CP2102 Pin outs
GND
TXD
RXD

Potentiometer Pin outs	
GND	GND
Output	Signal
VCC	3.3 V DC

Voltage Divider Pin outs	
GND	GND
3.3V	Signal
5 V	5 V DC

Set drive to Analog Control Of Motor in Software:

- By default the drive is in Analog mode. However if the mode is changed to some other then it can be set through software.
- Connect drive to PC and select **Analog Control Mode** in Operation Mode

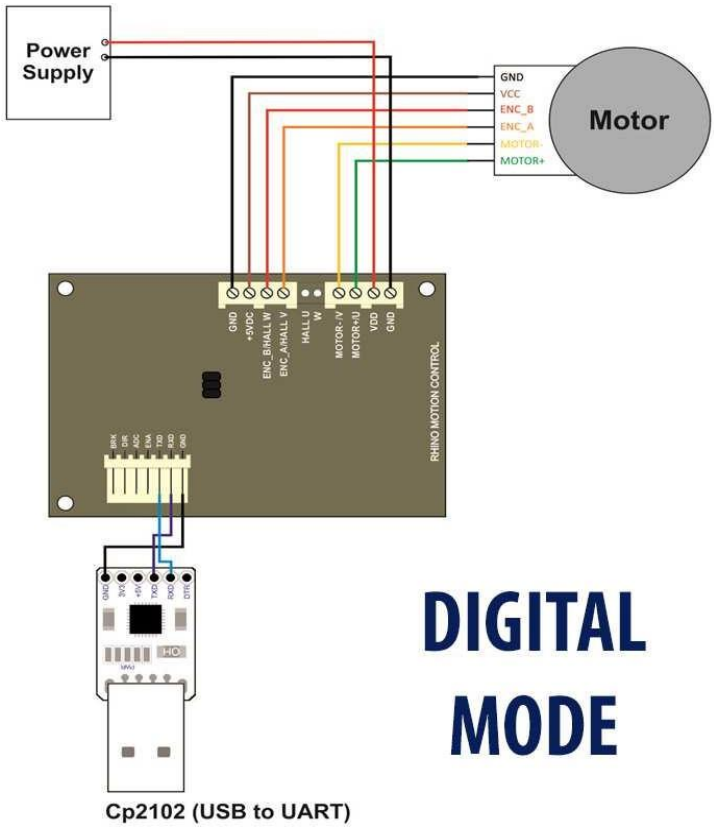


- After selecting analog control mode click on "Write Parameters" to always start the drive in analog mode.

Mode 1 - Digital Speed Control Mode

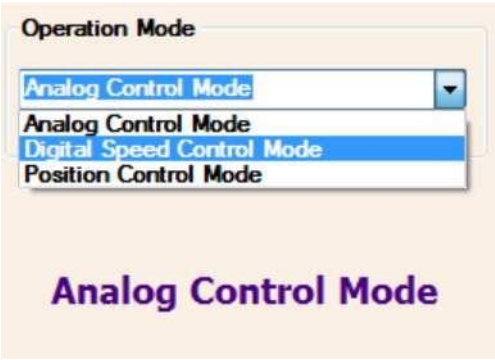
- In this mode the drive works in speed mode. It tries to reach and maintain the speed as per set parameter.
- The speed can vary depending upon load and gains set.
- Make sure that the set speed is always less than the motor can actually reach, otherwise the drive can give uncertain result. For example if the maximum speed motor can reach at given voltage is 18000 always set speed less than 18000 RPM.

Hardware Connection:

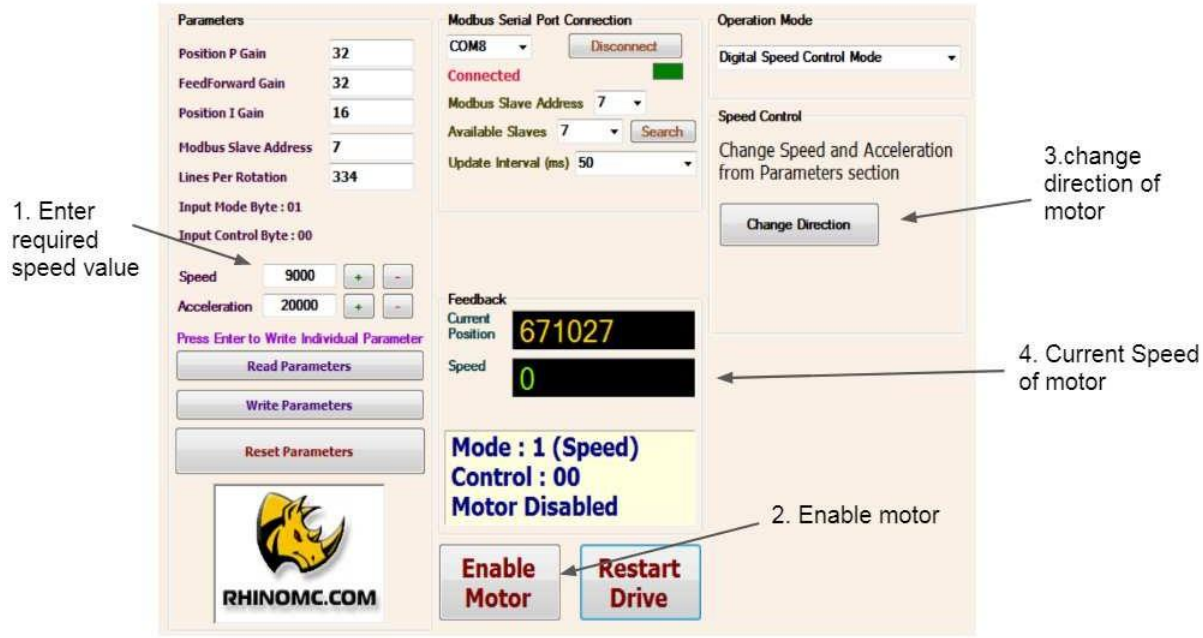


Digital Control Mode in Software:

a). To run your motor on digital speed control mode select this mode from operation mode.



b) In this mode speed of motor(rpm) can be controlled by changing value from 0-65535.



1. Speed can be increased and decreased using "+" and "-" respectively.

2. Enable Motor : Once speed is set click on "Enable Motor" to run motor on set speed.

3. Change Direction: Change the direction of motor. Can be done when motor is running or stopped.

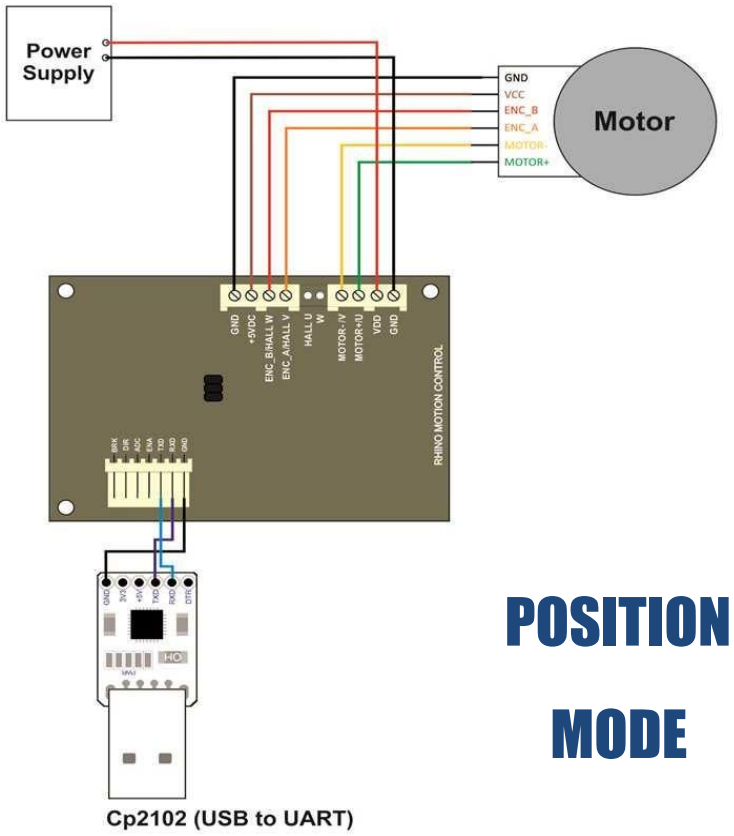
4.Speed Feedback: Current speed feedback from drive. This depends on actual speed measured by encoder and Lines per rotation setting.

Mode 2 - Position Control Mode

The position control mode allows to go to absolute position with reference to 0 as start position. Current position can be set to 0 at any point of time. The drive also manages the speed and acceleration/deceleration as per set parameter while responding to any change in set position.

When a 32bit value is set through register 16 and 18 the drive will try to move motor to that position. It can be updated even when motor is moving. Motor can be stopped any time or speed can be changed anytime even while moving.

Hardware Connection:



COUNT CALCULATION:

Lines per rotation concept is very important for Position Control mode. For example we will use HIGH TORQUE HIGH PRECISION ENCODER DC GEARED MOTOR 12V 200RPM (RMCS-5014)
https://robokits.co.in/motors/encoder-dc-servo/high-torque-high-precision-encoder-dc-geared-motor-12v-200rpm

Lines per rotation = 334

For one rotation of encoder, the no of counts = 334 x 4 = 1336 (As it is quad encoder) So for 1336 steps, the base motor would move one rotation.

However as the motor is geared motor, the ratio of the gear box determines the rotation of the output shaft of the motor for each rotation of the base motor.

output shaft(rpm) = base motor(rpm) / gear ratio

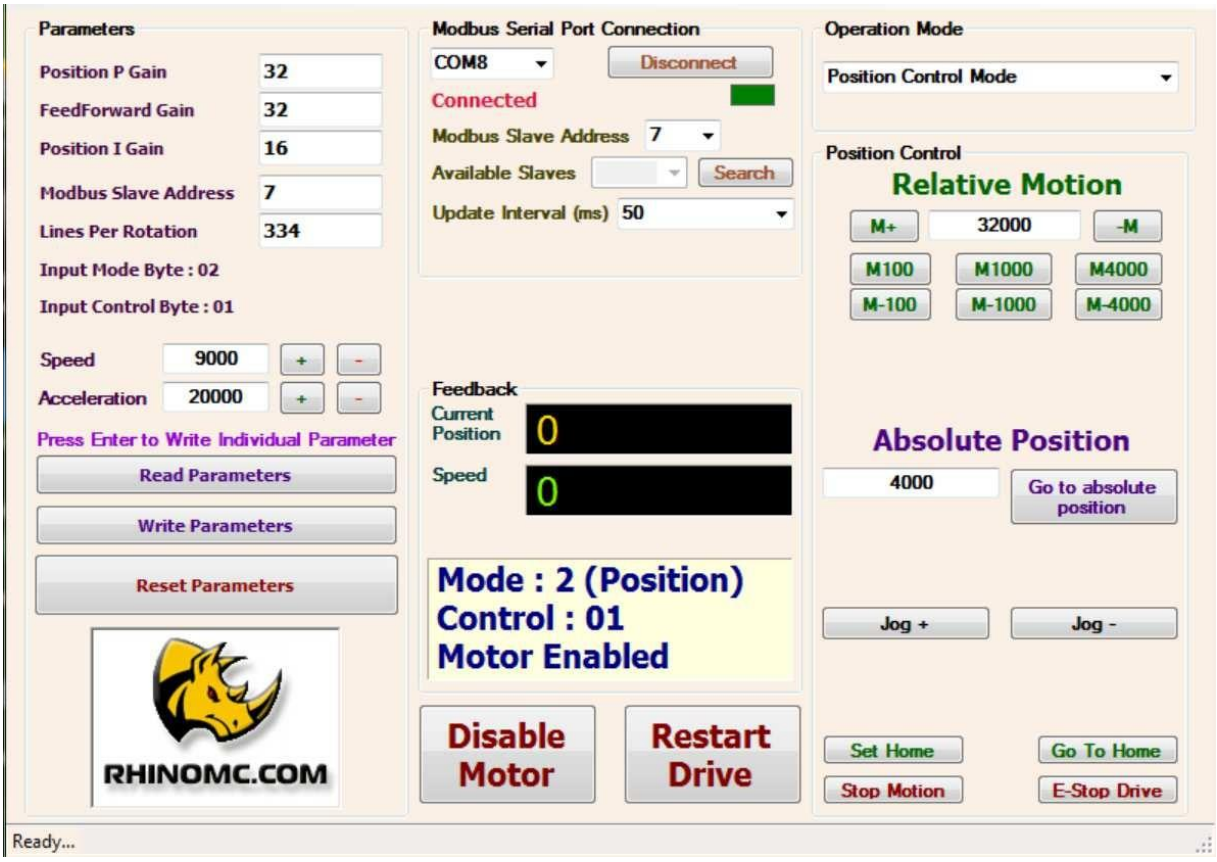
counts per rotation = counts per rotation X gear ratio for output shaft of base motor

For example, A motor of 200 RPM with base motor of 18000 rpm, the gear ratio is 90. Hence for a full rotation of the output shaft in this motor, the number of counts to be programmed would be 1336 x 90 = 120,240 steps.

This means output shaft will complete one rotation when Hex value of 120240 counts is set to register 16 and 18, LSB and MSB respectively.

Position Control in Software

When drive is connected and put in Position Control mode, software will look like this.

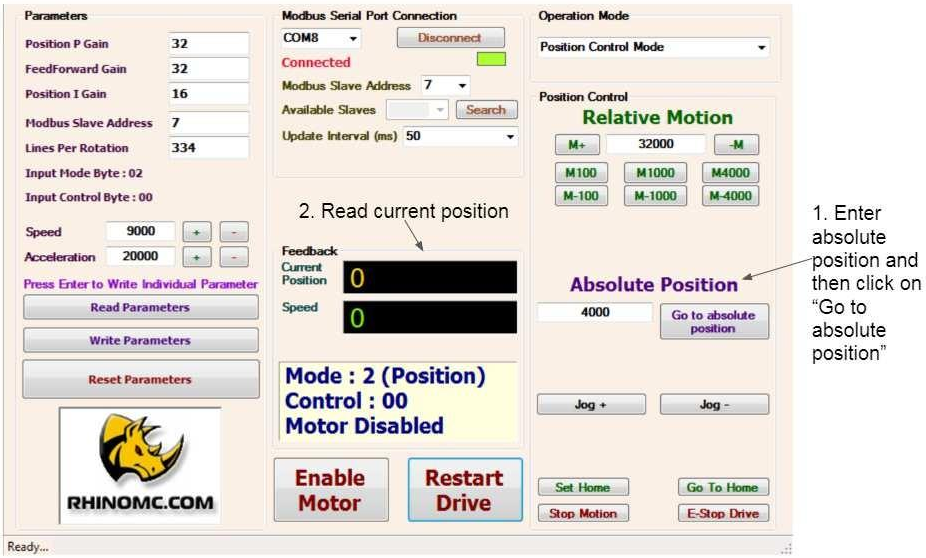


In Position control mode there are two ways to control position: **Absolute Position control** and **Relative Position control**.

ABSOLUTE POSITION

In this control mode motor will move precisely until entered count is not obtained. Position command can be given by putting number in Absolute Position textbox and clicking 'Go to absolute position' button. The direction is automatically managed by the drive. For example writing 4000 and clicking button moves the motor forward to 4000 count, giving -4000 would move the motor in other direction to achieve -4000 count. Feedback section

shows the actual current position and speed of motor.



RELATIVE POSITION

Relative position is a software function, the software takes current position as reference and moves motor to the count typed in textbox with reference to current position.

For example, if current position is 5000 and entered value is 3000 in forward direction (M+) then motor will move to 8000 position (absolute) and then if direction is reverse (M-) then it will move to 2000 position (absolute).

There are also some preset command buttons to move motor. Any of these buttons will not function if motor is moving (speed is not 0).

Other commands in Position mode

"Jog+" and "Jog-" are used to move motor manually in forward and reverse direction. It can be used when motor needs to be manually moved to some location like for homing.

"Set Home": To set current position as home position - 0. "Go to Home": Go to position 0.

"Stop Motion": Stop motor but keep position loop on. Motor stops but holds the current position. "E-stop Drive": By clicking this button drive will cut-off power supply to motor and motor gets free.

Modbus Registers

Register	Data Address (Decimal)	Access	Size	Function	Range / Command And Default value in HEX (Decimal)	Specification	Description
40001	0	Read / Write	16 bit	Write Parameters to EEPROM	FF (255)	For slave is 7 Mode byte:07 Control byte: FF	To save parameters in Drive (EEPROM) send Hex value XXFF to Address 0. Where XX is **slave ID** ranging from 01 to F7(1 - 247) *For example:* *If slave id is 7 then write 07FF.*
				Modbus Slave Address	Range:1-F7 (1-247) Default: 0B(11)		For Slave Address Setting refer **Slave ID Addressing on page 7**
40003	2	Read / Write	16 bit	**Modes (Default value: 0)**			
				Analog Control Mode **(Mode 0)**	0001(1)	Mode byte:00 Control byte:01	Enable motor in analog Control mode. For analog mode, connect external potentiometer.
					0000(0)	Mode byte:00 Control byte:00	Disable motor in analog Control mode
				Digital Speed Control Mode **(Mode 1)**	0101(257)	Mode byte:01 Control byte:01	Enable motor in CW (Clockwise Direction)
					0100(256)	Mode byte:01 Control byte:00	Disable motor in CW
					0104(260)	Mode byte:01 Control byte:04	Brake in CW
					0109(265)	Mode byte:01 Control byte:09	Enable motor in CCW (Counter Clockwise Direction)
					0108(264)	Mode byte:01 Control byte:08	Disable motor in CCW
					010C(268)	Mode byte:01 Control byte:0C	Brake in CCW
				Position Control Mode **(Mode 2)**	0201(513)	Mode byte:02 Control byte:01	Enable motor in position control mode
					0200(512)	Mode byte:02 Control byte:00	Disable motor in position control mode
				Special Functions			
				E-stop **(Mode 7)**	0700(1792)	Mode byte:07 Control byte:00	Stops motion and stops giving power to motor
				Stop **(Mode 7)**	0701 (1793)	Mode byte:07 Control byte:01	Stops motion, motor maintains the position
				Set Home Position **(Mode 8)**	0800(2048)	Mode byte:08 Control byte:00	Set current encoder count to 0

				Restart Drive **(Mode 9)**	0900(2304)	Mode byte:09 Control byte:00	Restarts the drive
40005	4	Read / Write	16 bit	Position Proportional Gain	Range: 01-FF (1-255) Default: 20(32)		Writing 0 in this address, then save to eeprom, on power reset the drive will load default values
40007	6	Read / Write	16 bit	Position Integral Gain	Range: 00-FF (0-255) Default: 10(16)		
40009	8	Read / Write	16 bit	Velocity Feed forward Gain	Range: 0-FF (0-255) Default: 20 (32)		
40011	10	Read / Write	16 bit	Lines per Rotation	Range: 0000-FFFF (0-65535) Default: 014E (334)		Set number of lines of encoder depends on encoder. Speed feedback depends on this.
40013	12	Read / Write	16 bit	Acceleration	Range: 0000 – FFFF (0-65535) Default: 4E20(20000)		Set the Acceleration of the Motor
40015	14	Read / Write	16 bit	Speed Command for base Motor in RPM	Range: 0000-FFFF (0-65535) Default:0800 (2048)		If gearbox is attached output RPM will be different.
40017	16	Read/ Write	32 bit	LSB of Position Command	Range: 0000-FFFF (0-65535) Default:0		Send Position Command, motion is affected only when register 18 is updated.
40019	18	Read / Write		MSB of Position Command	Range: 0000-FFFF (0-65535) Default:0		
40021	20	Read Only	32 bit	LSB of Position Feedback	Range: 0000-FFFF (0-65535) Default:0		Position Feedback Register
40023	22	Read Only		MSB of Position Feedback	Range: 0000-FFFF (0-65535) Default:0		
40025	24	Read Only	16 bit	Speed Feedback	Range: 0000-FFFF (0-65535) Default:0		Current Speed Feedback

Modbus Register Mapping:

- **Device Modbus Address(MOD_ID)**

  Address: 0x00 (40001)
  Default value: 0x0BFF

MOD_ID[15:8]								SP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 **MOD_ID[15:8]:** Modbus address bits
Default: 0x0B
Maximum value: 0xF7
Minimum value: 0x01
Bits 7:0 **SP[7:0]:** Save parameters to eeprom
Default: 0x00
If slave id is 7 then write 07FF

- **Control and Mode Register**

  Address: 0x02(40003)
  Default value: 0x0000

MODE[15:8]								CRTL[7:0]							
0	0	0	0	rw	rw	rw	rw	0	0	0	0	DIR	BRK	0	EN

Bits 15:8 **MODE [15:8]:** Mode byte
Default: 0x00

Bits [11:8] **0000**: Analog Control Mode
0001: Digital Control mode
0010: position Control Mode
0111: Stop mode
1000: Set Home Position
1001: Restart Drive

Bits 7:0 **CRTL [7:0]:** Control Byte
Default: 0x00

Bit 3 **DIR:** Direction (only Digital mode)
0: Clockwise direction
1: Counter-clockwise direction

Bit 2 **BRK**: Brake
0: Disable Brake
1: Enable Brake
Note: This function is only available in **Digital control mode.**

Bit 0 **EN**: Enable bit
0: Disable mode
1: Enable mode

Note: Enable has more priority than brake.

- **Position Proportional Gain Register (PPG)**

  Address: 0x04 (40005)
  Reset Value: 0xFF20

Reserved								PPG[7:0]							
1	1	1	1	1	1	1	1	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved
Bits 7:0 **PPG [7:0]**: Position Proportional Gain register
Default: 0x20
Maximum value: 0xFF
Minimum value: 0x01

Position proportional gain provides stable and vibration less movement in motor.

- **Position Integral Gain Register (PIG)**

  Address: 0x0006 (40007)
  Reset Value: 0xFF10

Reserved								PIG[7:0]							
1	1	1	1	1	1	1	1	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved.
Bits 7:0 **PIG [7:0]:** Position Integral Gain register
Default: 0x10
Maximum value: 0xFF
Minimum value: 0x00
Position Integral gain affects torque of the motor.

- **Velocity Feed forward Gain Register (VFFG)**

  Address: 0x08 (40009)
  Reset Value: 0xFF20

Reserved								VFFG[7:0]							
1	1	1	1	1	1	1	1	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved.
Bits 7:0 **VFFG [7:0]**: Velocity Feed forward Gain register
Default: 0x20
Maximum value: 0xFF
Minimum value: 0x00

Velocity feed-forward gain predicts the commands needed to achieve zero error and inject them into the control loop.

- **Lines per Rotation Register (LPR)**

  Address: 0x0A (40011)
  Reset value: 0x014E

LPR[15:0]
rw

Bits 15:0 **LPR [15:0]:** Lines per Rotation
Default: 0x014E
Maximum value: 0xFFFF
Minimum value: 0x0000

This register is allocated to set number of lines of an Encoder of the Motor. The speed feedback and position feedback depends on this register.

- **Acceleration Register(ACC)**

Address: 0x0C (40013)
Reset value: 0x4E20

ACC[15:0]
rw

Bits 15:0 **ACC [15:0]:** Acceleration
Default: 0x4E20
Maximum value: 0xFFFF
Minimum value: 0x0000

Acceleration is applied in speed and position control modes when starting the motion or changing the speed. Deceleration is also applied as per set value when stopping the motor or decreasing the speed.

- **Speed Register(SPD)**

Address: 0x0E (40015)
Reset value: 0x0800

SPD[15:0]
rw

Bits 15:0 **SPD [15:0]:** base motor speed in RPM
Default: 0x0800
Maximum value: 0xFFFF
Minimum value: 0x0000

If gearbox is attached output RPM will be different.

- **LSB of Position Command (LSB_POS)**

Address: 0x0010 (40017)
Reset value: 0x0000

LSB_POS[15:0]
rw

Bits 15:0 **LSB_POS [15:0]:** LSB of Position Command
Default: 0x0000
Maximum value: 0xFFFF
Minimum value: 0x0000

- **MSB of Position Command (MSB_POS)**

  Address: 0x12 (40019)
  Reset value: 0x0000

MSB _POS[15:0]
rw

Bits 15:0 **MSB_POS [15:0]:** MSB of Position Command
Default: 0x0000
Maximum value: 0xFFFF
Minimum value: 0x0000

Motor will move only when register 18 is updated.

- **LSB of Position Feedback (LSB _POS_FB)**

  Address: 0x14 (40021)
  Reset value: 0x0000

LSB _POS_FB[15:0]
r

Bits 15:0 **LSB_POS [15:0]:** LSB of Position Feedback
Default: 0x0000
Maximum value: 0xFFFF
Minimum value: 0x0000

- **MSB of Position Feedback (MSB _POS_FB**)

  Address: 0x16 (40023)
  Reset value: 0x0000

MSB _POS_FB[15:0]
r

Bits 15:0 **MSB_POS [15:0]:** MSB of Position Feedback
Default: 0x0000
Maximum value: 0xFFFF
Minimum value: 0x0000

- **Speed Feedback (SPD_FB)**

  Address: 0x18 (40025)
  Reset value: 0x0000

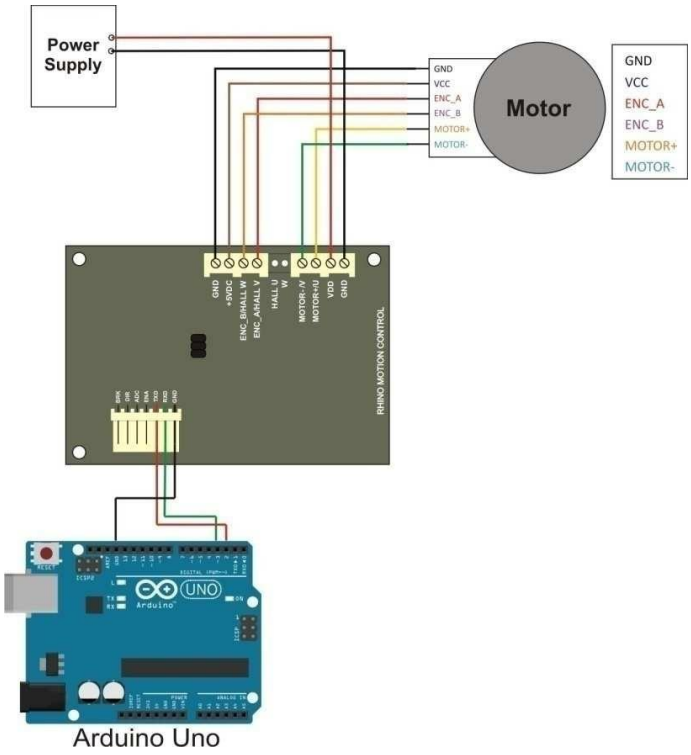
SPD_FB[15:0]
r

Bits 15:0 **SPD_FB [15:0]:** Speed Feedback of base motor in RPM
Default: 0x0000
Maximum value: 0xFFFF
Minimum value: 0x0000
Read Current Speed of the motor in RPM. If gearbox is attached then output RPM will be different.

Control through Arduino Microcontroller

RMCS2303 drive can be controlled by any microcontroller, PLC or PC software capable of communicating on ASCII Modbus protocol. Following section shows how to use provided library for Arduino microcontroller.

Hardware Connection:



Arduino Uno

DRIVE-ARDUINO MEGA CONNECTION

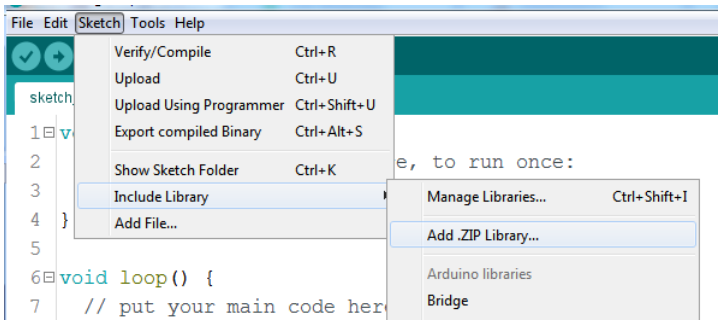
DRIVE	ARDUINO
GND	GND
RXD	TXn (n=1,2,3 for Arduino mega2560)
TXD	RXn (n=1,2,3 for Arduino mega2560)

DRIVE-ARDUINO UNO CONNECTION

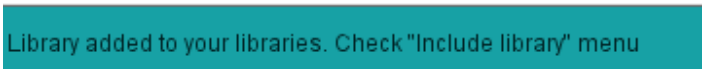
DRIVE	ARDUINO
GND	GND
RXD	D3(Software Serial Tx)
TXD	D2(Software Serial Rx)

Library Installation:

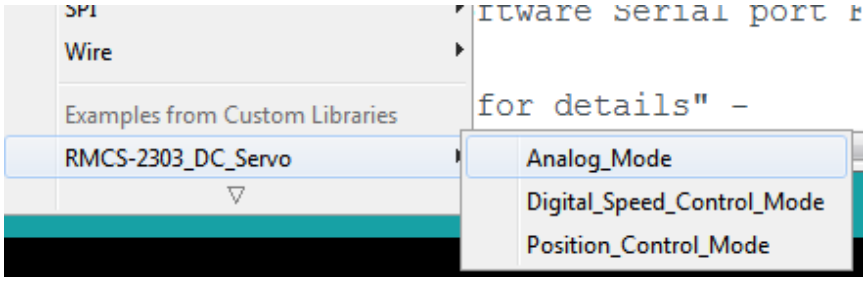
- Download RMCS-2303 Arduino library from : http://robokits.co.in/downloads/RMCS2303drive_Arduino_Library.zip
- In Arduino IDE got to menu - Sketch > Include Library > Add .ZIP Library



- Choose the downloaded zip file.
- You can see confirmation if library import is successful



- Open examples from Menu File > Examples > RMCS-2303_DC_Servo



- Set parameters; make sure you choose parameters, serial port, and slave id properly before uploading the code to Arduino board.
- There are many more functions other than covered in example codes. Description of these functions as below. In this code 'rmcs' is object for library class, it can be changed.

Functions for analog control mode

rmcs.Enable_Analog_Mode(slave_id); //Enable analog mode

rmcs.Disable_Analog_Mode(slave_id); //Disable motor in analog mode

Functions for Digital speed control mode

rmcs.Speed(slave_id,5000); //Set speed to 5000 RPM

rmcs.Enable_Digital_Mode(slave_id,1); //Enable speed mode, motor starts moving long int b=rmcs.Speed_Feedback(slave_id);//Speed feedback from drive

rmcs.Brake_Motor(slave_id, 1); //Brake motor in speed mode

rmcs.Disable_Digital_Mode(slave_id,1); //Disable motor in speed mode

```
Functions for position control mode
rmcs.Absolute_position(slave_id,-10000);    //Move to absolute position -
10000 long int b=rmcs.Position_Feedback(slave_id);//Take position
feedback from drive Serial.println(b);

Special Functions
rmcs.SAVE(slave_id);     //Save all parameters
rmcs.RESET(slave_id);   //Reset drive to default
parameters, rmcs.ESTOP(slave_id);          //E-stop the
drive rmcs.STOP(slave_id);          //Stop motion
rmcs.SET_HOME(slave_id);          //Set current position to
0 rmcs.Restart(slave_id);              //Software restart drive.
```

Description of code and functions used

```
#include<RMCS2303drive.h>
```

Include Downloaded library in code to use available functions for controlling RMCS2303 drive in all modes. If

Library is not installed then follows steps as per described in “Library Installation".

```
RMCS2303 rmcs;
```

Make an object for class RMCS2303. Any object name can be set instead of "rmcs". Use the same class in

all functions.

```
byte slave_id=7; int INP_CONTROL_MODE=265; int PP_gain=32; int PI_gain=16; int VF_gain=32;
int LPR=334; int acceleration=20000; int speed=320;
```

Set all necessary parameters value as per application. INP_CONTROL_MODE depends on Mode and type

of control, values are given in Hex in Appendix 1 so convert it in integer then store in INP_CONTROL_MODE variable. LPR is Lines Per Rotation.

```
rmcs.Serial_selection(1);          //Serial portselection:0-Hardware serial,1-Software serial
```

Set hardware or software serial port to be used. Boards like Arduino Uno have only one hardware serial port

this gets used by usb-serial. In this case software serial should be used to communicate with drive.

Boards like Arduino mega have multiple hardware serial ports, in this case use hardware serial.

Hardware Serial port: rmcs.Serial_selection (0); Software Serial Port: rmcs.Serial_selection (1);

rmcs.Serial0(9600); //usb serial to monitor data on serial monitor

This function will initialize Serial communication by setting baudrate between Arduino and Serial monitor in Arduino IDE. Baudrate can be changed as required.

rmcs.begin(&Serial3,9600); //uncomment if using hardware serial port for mega2560:Serial1,Serial2,Serial3 and set baudrate

This function passes the pointer of serial port to be used. It's used to initialize serial port and set baudrate.

Serial1 : rmcs.begin(&Serial1,9600); Serial2 : rmcs.begin(&Serial2,9600); Serial3 : rmcs.begin(&Serial3,9600);

rmcs.begin(&myserial,9600); //uncomment for using software serial and set baudrate

This function is used to set baudrate of Software Serial port. In Arduino Uno software serial should be used.

rmcs.WRITE_PARAMETER(slave_id, INP_CONTROL_MODE, PP_gain, PI_gain, VF_gain, LPR, acceleration, speed);

This function is used to write all parameters. Confirmation will be sent on Serial port shown as below when

all parameters are written Successfully.

```
INP_CONTROL :          DONE
INP_MODE :             DONE
PP_GAIN :              DONE
PI_GAIN :              DONE
VF_GAIN :              DONE
LPR :                  DONE
ACCELERATION :         DONE
SPEED :                DONE
```

rmcs.READ_PARAMETER(slave_id);

This function read all current parameters and then print on Serial monitor.

Analog Mode:

rmcs.Enable_Analog_Mode(slave_id);

This function will enable your drive in Analog control mode. Speed of motor can be changed by varying

potentiometer.

rmcs.Disable_Analog_Mode(slave_id); // To disable motor in Analog control Mode

This function is used to disable motor in analog control mode.

Digital Speed Control Mode:

rmcs.Speed(slave_id,5000); //enter speed within range of 0-65535

When drive is used in Digital Speed control mode, this function is used to change the speed. Value from 0

to 65535 can be set but all motors have some maximum rpm. When value is more than possible RPM motor will run at full speed, this may cause jerks in motion commands.

rmcs.Enable_Digital_Mode(slave_id,1);

Enable motor in digital mode with forward or reverse direction. 0 is forward (encoder count +) and 1 is reverse (encoder count -).

Forward : rmcs.Enable_Digital_Mode(slave_id,0); Reverse: rmcs.Enable_Digital_Mode(slave_id,1);

rmcs.Speed_Feedback(slave_id);

This function returns the current speed of motor in long integer. Speed less than 0 is received when motor is moving in reverse direction.

rmcs.Brake_Motor(slave_id,1);

This function is used to brake motor in digital speed control mode. Should be used with direction as below, if not drive may misbehave and motor may jerk.

Forward: rmcs.Brake_Motor(slave_id,0); Reverse: rmcs.Brake_Motor(slave_id,1);

rmcs.Disable_Digital_Mode(slave_id,1);

This function will disable Digital speed control mode in entered direction. Should be used with direction as below, if not drive may misbehave and motor may jerk.

Forward: rmcs.Disable_Digital_Mode(slave_id,0); Reverse: rmcs.Disable_Digital_Mode(slave_id,1);

Position Control Mode:

rmcs.Absolute_position(slave_id,10000); This function enables drive in position control mode. Positive range is 1 to 2147483647 and negative range is -1 to -2147483648. Direction is selected automatically by drive.

rmcs.Absolute_position(slave_id,-10000);

rmcs.Position_Feedback(slave_id);

Current Position of motor can be read by this function on Serial monitor. This function will return long integer.

Special Functions:

rmcs.SAVE(slave_id);

This function will save all written parameters in drive.

rmcs.RESET(slave_id);

This function will reset all the parameters and set all parameters at default value.

rmcs.ESTOP(slave_id);

This function will stop motor and cut off power from drive to motor.

rmcs.STOP(slave_id);

This function is used to stop motion of motor. Motor will be locked as its still controlled by drive.

rmcs.SET_HOME(slave_id);

This function changes current encoder position to "0".

rmcs.Restart(slave_id);

Restart the drive by software.