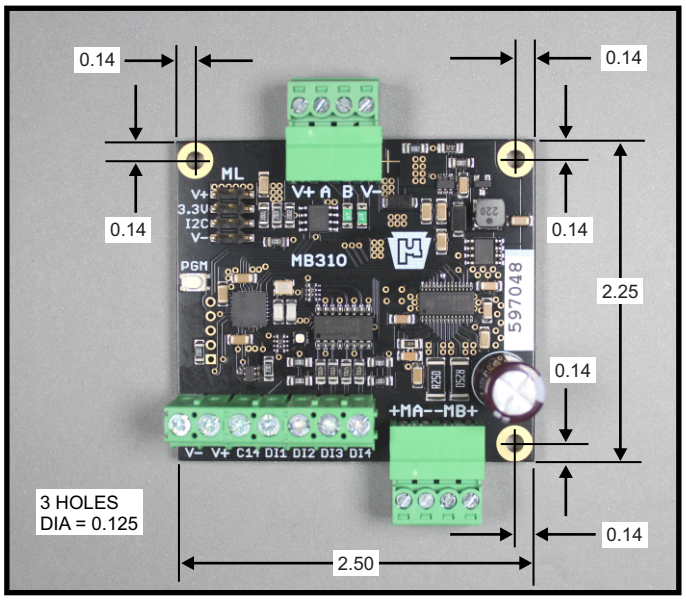


Description

Microva model MB310 is a bipolar (4 wire) stepper motor controller and driver with up to 2.5A per phase. The MB310 is fully programmable using Microva's free BASIC compiler and can operate "stand alone" with no separate controller. The MB310 includes RS-485 I/O and supports Modbus RTU communication with up to 247 nodes on the network. The MB310 is factory programmed to be a Modbus RTU slave peripheral device and includes a motor control scripting language interpreter that makes it easy to control a stepper motor with minimal programming. The MB310 includes four opto-isolated digital inputs and the Microva ML I/O port for I2C communication with external devices made by Microva or other manufacturers. The ML port makes it easy to connect to peripheral devices using standard 0.1 inch pitch connectors and ribbon cable.

Power Supply

The V+ and V- input pins provide power to the PCB and to the motor. The MB310 includes an industrial range switch mode power supply. All V+ terminals on the PCB are shorted together and all V- terminals are shorted together. V+ and V- are also used to power the motor, therefore, the power supply should be located near the PCB due to the high motor drive current. The 3.3V pin on the ML I/O port is used to power external I/O PCBs and is an output only. Table 3 shows the power supply specifications.

How To Load A New Program

The MB310 must be programmed from the RS-485 I/O port. The Microva MB100 USB-to-RS485 interface is required for programming. Program mode can be started from the computer, as explained in the Microva BASIC Programming Reference Manual, or by holding the PGM switch and turning on the power.

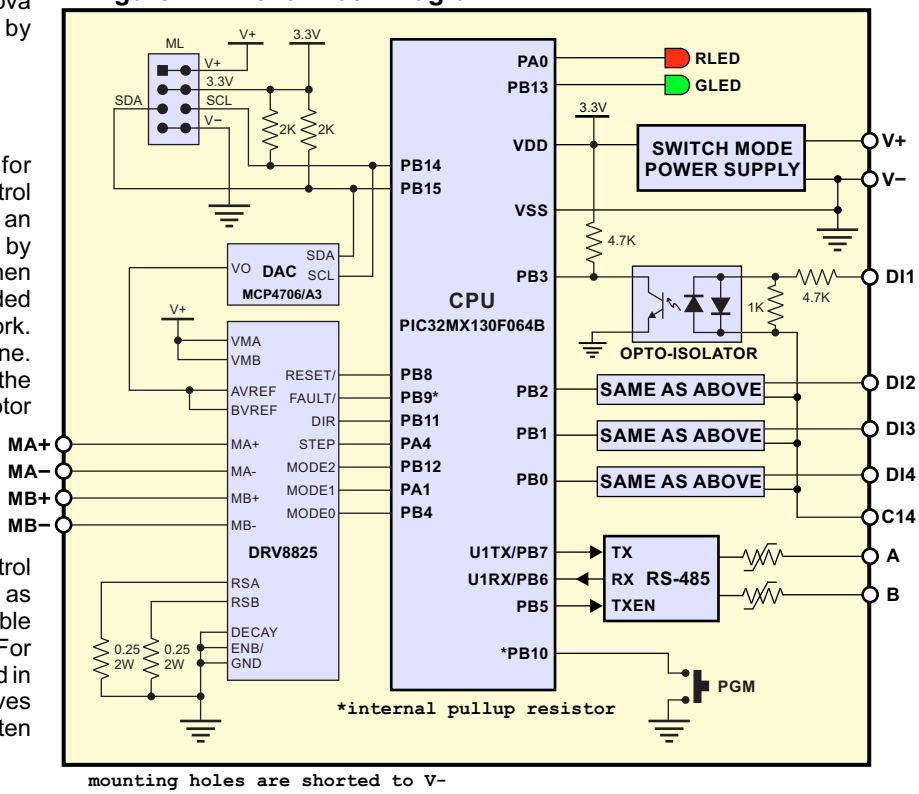
The Motor Control Scripting Language

The MB310 comes preloaded with software for controlling the motor using a simple motor control scripting language. The motor control script is just an ordinary table of integer data that is loaded at reset by the application program into RAM memory and then executed. The motor control script can also be loaded by the Modbus master over the RS-485 network. Commands in the script are executed line-by-line. Once a command line is completed the next line in the script is executed. For example, to move the motor forward 100 steps the script commands are:

```
WriteVar, 100, Offset
MoveOffset, Fwd, 0, 0
```

Up to 100 variables can be used in the motor control script. Several of these variables are predefined, as shown in Table 1, but most are undefined and available for any purpose required in the motor control script. For example, variable MotorSpeed sets the motor speed in pulses per second (pps) for any command that moves the motor. All the variables can be read and written over the Modbus by the Modbus master.

Figure 1: MB310 Block Diagram



The Status Variable

The Status variable is constantly updated in the program and holds the state of several status bits. The status bits are:

b0: the state of the PGM switch on the PCB.
b1: the state of opto-isolator input DI1 (which is also the DI1Counter and DI1Frequency input).
b2: the state of opto-isolator input DI2
b3: the state of opto-isolator input DI3
b4: the state of opto-isolator input DI4 (which is the emergency stop input)
b5: the Fault state (Fault indicates the motor driver IC is over temperature or the motor coil is open or shorted)

Bits 6 thru 31 of Status can be used for any purpose in the motor script. The Modbus master can write to the Status variable to set or clear bits 6 thru 31 (any values written to bits 0 to 5 are ignored because the script control program over writes these bits with the states described above). Status bit 4 is the emergency stop input and bit 5 is the motor driver fault state. If either bit 4 or bit 5 turns on the script control program is immediately stopped, the motor driver IC is turned off, and the PCB red LED will blink continuously. Bits 4 and 5 are latching bits meaning once they turn on the only way to turn them off is to reset the program (or turn the power off/on). Figure 2 shows the method to connect open collector source or sink inputs to the opto-isolator inputs DI1 thru DI4.

The Condition State

Most of the script commands include a condition which must be true in order for the command to execute. For example, the command Move, Fwd, PGM, Off will drive the motor forward as long as the PGM switch on the PCB is off. When the condition if true, i.e. the PGM switch is off, the Move command will continue to execute and the motor will drive forward at the speed set in the MotorSpeed variable. As soon as the PGM switch turns on the command is ended and the next command in the control script is executed. The condition state is calculated by logically ANDing the condition bit mask with the Status variable, described above, and checking if the result is zero or not zero. For the example, the condition bit mask = PGM and the condition is OFF. PGM is a constant defined in the software = 1 (bit 0). Therefore, the calculation is: (1 AND Status). So if bit 0 of Status is off then the result is zero, therefore, the condition OFF is true. If bit 0 is on, i.e. if the PGM switch is on, then the result = 1 which is not zero so the condition OFF is false and the command will end. Several bits can be combined in one bit mask and the condition can be OFF (zero) or ON (meaning not zero). For example, in the command Move, Fwd, 0b1100, On the condition is true if either bit 2 OR bit 3 of Status is on. When bit 2 AND bit 3 are both off the result of (0b1100 And Status) is zero and the condition is no longer true. So the command can be thought of to mean "move forward while DI2 or DI3 is on". Many times the condition in the command is not needed in which case set the bit mask and the condition both to zero which makes the condition always true. For example, MoveOffset, Fwd, 0, 0 will move to offset, ignoring Status.

Motor Current and Motor Speed

The maximum current in each phase of the motor coil can be set by writing to the MotorCurrent variable in the motor control script program. For example, the command to set the coil current to 500mA in each phase is WriteVar, 500, MotorCurrent. This sets the VREF input to the DRV8825 motor driver IC as shown in the block diagram. The stepper motor torque drops off drastically as the motor speed increases due to the time it takes for the coil currents to change from minimum to maximum and back as the motor is stepped. Most stepper motors cannot step faster than around 1500 pulses per second at a full step size. The speed can be increased by using smaller step sizes. The DRV8825 motor driver IC can be set to step sizes as small as 1/32 of a full step using the MODE input pins as described in the data sheet. The MB310 application program maximum motor speed is approximately 25,000 pps. Every move command increments or decrements variable MotorPosn as the motor is stepped. When the motor is moving forward MotorPosn is incremented. When moving backward MotorPosn is decremented. The program assumes that each STEP input to the motor driver IC moves the motor one step, however, keep in mind that when the motor is moving and the motor direction is reversed, the rotor momentum may cause the motor to loose step and, therefore, be out of sync with MotorPosn. This only occurs when the speed is fast enough that the motor cannot reverse in one step pulse period.

The Motor Control Script Commands

Table 2 shows a summary of the motor control script commands. Array MotorScript holds the motor control script which can be loaded directly from a table in the MB310 program (see the examples at the end of this document) or the Modbus master can write the control script into the array when RunMode is off. ScriptX holds the script index to array MotorScript and, therefore, controls which command in the script is being executed. The jump commands overwrite ScriptX with a new location in MotorScript which changes the order of command execution. All ScriptX values in the motor control script are relative offsets from the start of the command. For example, the command Jump, 0, PGM, Off waits for the PGM switch on the PCB to turn on. The jump location is 0 which means the command repeats over and over until the PGM switch turns on at which point ScriptX is written with the index value for the next command in the script. As explained above, most commands include a bit mask and condition which must be true for the command to execute. Commands continue to execute until the condition is no longer true or the command completes. For example, the MoveSetPoint command will continue to move the motor, both forward and backward, until MotorPosn equals MotorSP then the command will end and ScriptX will be incremented to start the next command in the list. Many of the motor move commands use the pre-defined variables in Table 1. These variables must be initialized using the WriteVar or CopyVar commands before the motor move command is executed.

Table 1: Pre-Defined Motor Control Script Variables and Modbus Addresses

Name	Addr	Description
RunMode	*1	motor run mode 0 = stop the motor script, turn off the motor driver and LEDs 1 = run the motor script (this is the value at reset) 2 = pause the motor script (motor remains on)
ScriptX	*2	index to MotorScript holds the command currently being executed (set to 0 at reset)
MotorScript	210	array which holds the motor control script (array is 1000 integers)
MotorSpeed	10	motor speed in pulses per second (pps) used by all move commands (set to 100 at reset)
MotorSP	12	motor step position set point used in command MoveSetPoint
MotorStepSize	14	DRV8825 motor driver IC MODE input pins (b2:MODE2, b1:MODE1, b0:MODE0)
MotorEnable	16	DRV8825 motor driver IC RES/ input pin (turns the motor driver IC + coil current on/off)
MotorCurrent	18	sets the current in each phase of the motor coil (mA)
DI1SampleTime	20	DI1 sample time in milliseconds used to calculate DI1Frequency (set to 1000 at reset)
RGBLeds	22	LED states (b2:red LED, b1:green LED, b0:blue LED) (blue LED is not connected on MB310)
MotorTime	24	motor time in milliseconds used in commands MoveWait and MoveWaitRamp
WaitTime	26	wait time in milliseconds used in command Wait
Speed0	28	starting speed (pps) used in commands MoveOffsetRamp and MoveTimeRamp
Speed1	30	ending speed (pps) used in commands MoveOffsetRamp and MoveTimeRamp
Offset	32	offset data used in commands MoveOffset and MoveOffsetRamp
LoopCount	34	general purpose loop counter
Adc	36	A-to-D converter value used in command ML122
Adc0	38	A-to-D converter value used for calibration
Adc1	40	A-to-D converter value used for calibration
Min	42	minimum value used in command ReScale
Max	44	maximum value used in command ReScale
Data	46	any data
Status	90	status bits (constantly refreshed in the program) b0: the PGM switch on the PCB b1: opto-isolator input DI1 (which is also the DI1Counter and DI1Frequency input) b2: opto-isolator input DI2 b3: opto-isolator input DI3 b4: opto-isolator input DI4 (the emergency stop input) b5: Fault state (indicates motor driver IC is over temp or coil is open or shorted) b6 thru 31: unassigned, can be used for any purpose in the motor script
MotorPosn	92	live motor step position
DI1Counter	94	DI1 pulse counter (cleared at reset)
DI1Frequency	96	DI1 frequency x 10 uses DI1SampleTime (described above) (ex. 3546 = 354.6Hz)

* Variables RunMode and ScriptX are 16 bit integer values that can be written by the Modbus master but cannot be changed in the motor control script. All other variables are 32 bit integers that require two Modbus addresses each. The data is in "big endian" format, i.e. addr n = upper 16 bits, addr n + 1 = lower 16 bits.

Modbus Communication

The unit supports Modbus RTU two wire RS-485 half duplex serial communication on the four pin plug terminal block labeled V+, A, B, V-. The terminal marked A is the non-inverting data pin and the terminal marked B is the inverting data pin. The tables show the baud rates and data formats supported. The factory set communication values are:

Node Number = 1
Baud Rate = 250000
Data Format = 8/N/1

The devices' node number, baud rate and data format can be changed using the Microva BASIC Compiler software running on the computer (see *"Microva BASIC Programming Reference Manual"*) or by using the PGM switch on the PCB in which case no computer is needed, see the flowchart on the next page. For more information see Microva application notes:

AN130 What is Modbus?
AN132 Microva Modbus RTU Implementation
AN136 RS-485 Network Wiring

Red and Green LEDs

The PCB includes red and green LEDS. In RUN mode the green LED will flash briefly each time a Modbus packet, addressed to the node, is received. The LEDs can be controlled by writing to variable RGBLeds in the motor control script. In PGM mode the red LED will be on and the green LED will be off.

Baud Rates

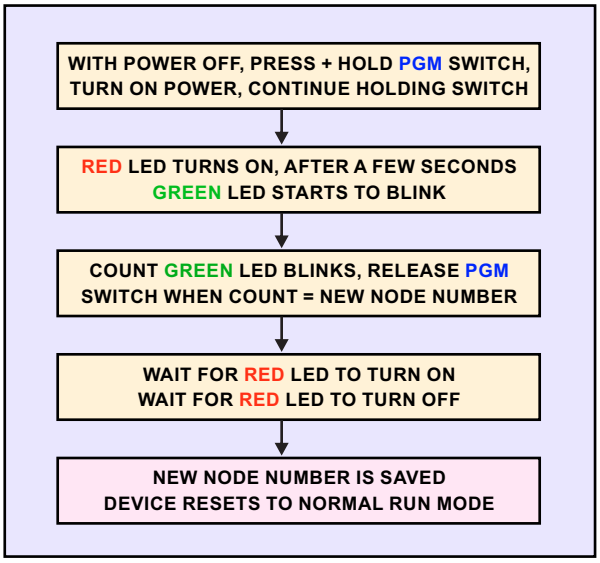
LED Count	Baud Rate
1	2400
2	4800
3	9600
4	19200
5	38400
6	57600
7	115200
8	250000
9	500000
10	1000000

Data Formats

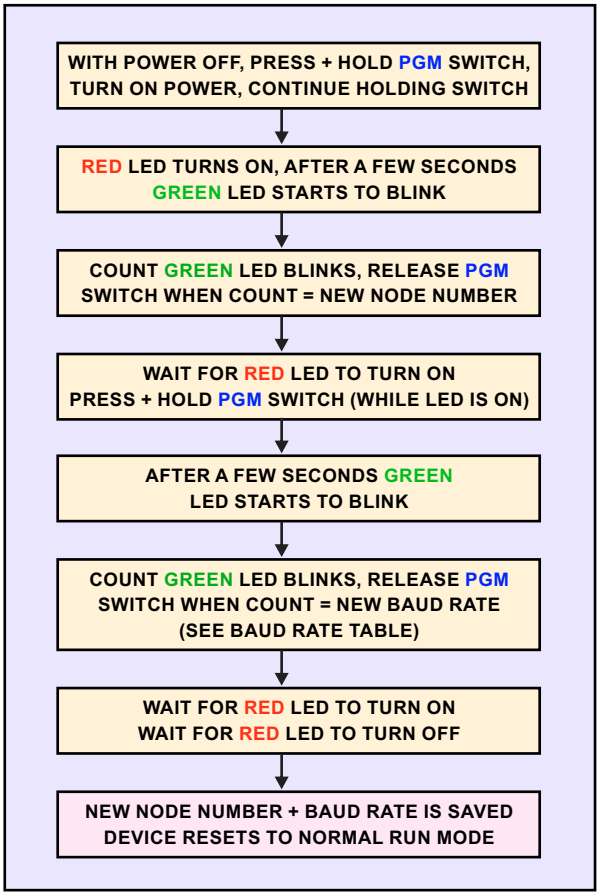
LED Count	Data Bits	Parity	Stop Bits
1	8	None	1
2	8	None	2
3	8	Even	1
4	8	Odd	1

Flowcharts to Change the Communication Settings Using the PGM Switch

Change Node Number



Change Node Number + Baud Rate



Change Node Number + Baud Rate + Data Format

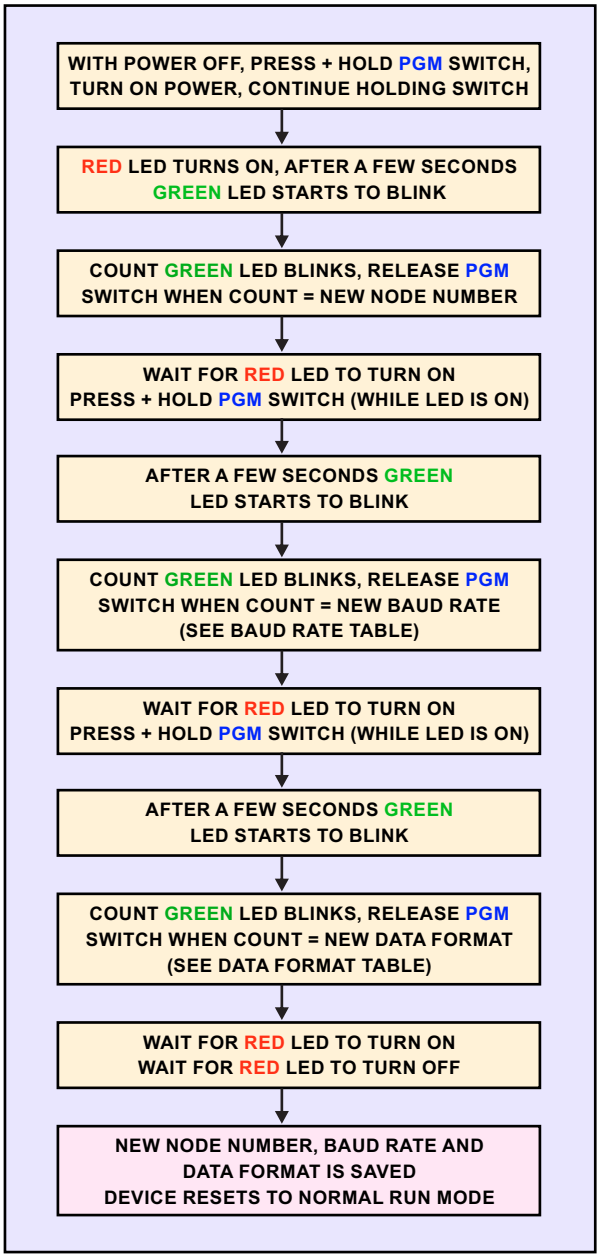


Table 2: Motor Control Script Commands

CompJump, Jump Location, Var1, Condition, Var2
Variable Var1 is compared with Var2 using Condition. ScriptX is set to Jump Location if the comparison is true. There are six possible conditions used in the comparision. The conditions are:
EQ (equal), NE (not equal), LT (less than), LE (less than or equal), GT (greater than), GE (greater than or equal)

CopyVar, SourceVar, DestinationVar
Copies the source variable to the destination variable. The motor driver IC, hardware and interrupt are updated if the destination variable is any of the following: MotorSpeed, MotorPosn, MotorStepSize, MotorEnable, MotorCurrent, RGBLeds

DecrJumpNZ, DecrVar, Jump Location
Variable DecrVar is decremented, if the result is not zero ScriptX is set to Jump Location.

DecrJumpZ, DecrVar, Jump Location
Variable DecrVar is decremented, if the result is zero ScriptX is set to Jump Location.

Jump, Jump Location, Condition Bits, Condition
If the condition is true ScriptX is set to Jump Location.

Math, Operation, SourceVar, DestinationVar
Variable DestinationVar is set = DestinationVar (Operation) SourceVar. The operations are:
Add (add), Subt (subtract), Mult (multiply), Div (divide), LAnd (boolean logic AND), LOr (boolean logic OR)

Move, Direction, Condition Bits, Condition
Moves the motor forward (FWD) or backward (BWD) as long as the condition is true.

MoveOffset, Direction, Condition Bits, Condition
While the condition is true the motor moves from MotorPosn to (MotorPosn +/- Offset).

MoveOffsetRamp, Direction, Condition Bits, Condition
Used to accelerate or decelerate the motor speed. While the condition is true the motor moves from MotorPosn to (MotorPosn +/- Offset) starting with motor speed Speed0 and finishing with motor speed Speed1.

MoveSetPoint, Condition Bits, Condition
Moves forward or backward to the motor position set point in MotorSP as long as the condition is true.

MoveTime, Direction, Condition Bits, Condition
Moves the motor forward (FWD) or backward (BWD) for the time period in MotorTime as long as the condition is true.

MoveTimeRamp, Direction, Condition Bits, Condition
Used to accelerate or decelerate the motor speed. Moves the motor forward (FWD) or backward (BWD) for the time period in MotorTime as long as the condition is true. The motor speed starts with Speed0 at time 0 and finishes with Speed1 at the time value in MotorTime.

ReScale, X, Y, X0, Y0, X1, Y1
Calculates a new Y value using the standard linear interpolation: Y = (X - X0) * (Y1 - Y0) / (X1 - X0) + Y0
The Y result is then checked against the values in variables Min and Max. If Y is less than Min then it is set to Min. If Y is greater than Max then it is set to Max.

Wait
The program pauses for the time period in WaitTime.

WriteVar, Data, DestinationVar
Data is the value to write to the destination variable. Data is not a variable. The motor driver IC, hardware and interrupt are updated if the destination is any of the following: MotorSpeed, MotorPosn, MotorStepSize, MotorEnable, MotorCurrent, RGBLeds

ML102, Address, SourceVar
Writes to Microva ML102 module with I2C address = Address. Data (b1=R1, b2=R2) is in SourceVar.

ML104, Address, SourceVar
Writes to Microva ML104 module with I2C address = Address. Data (b1=R1, b2=R2, b3=R3, b4=R4) is in SourceVar.

ML122, Address, Channel, ResultVar
Reads the Microva ML122 dual channel 4/20mA input module with I2C address = Address. The channel read = Channel (must be 1 or 2). The result is placed in variable ResultVar.

Figure 2: Open Collector Source/Sink Inputs

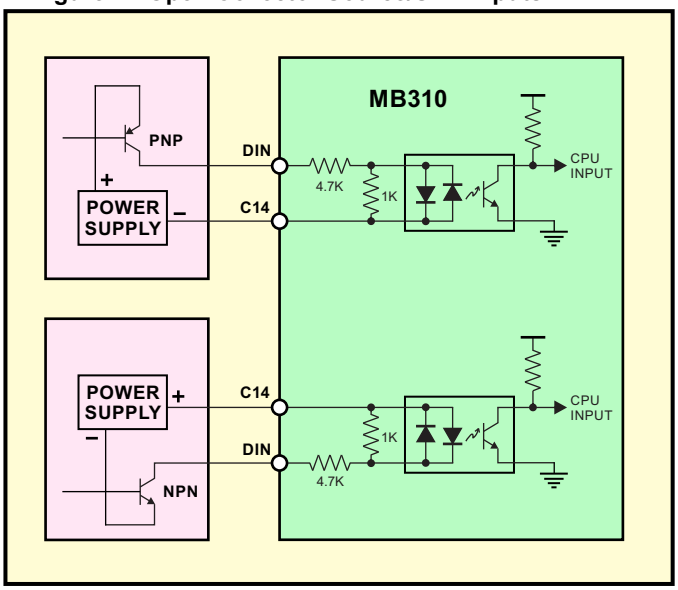


Table 3: MB310 Specifications

Power Supply Voltage (V+)	8.5V-30V
Typical Operating Current (1)	20mA(12V), 14mA(24V)
3.3V Max Output Current (2)	350mA
Max Voltage On Any I/O Pin (3)	3.6V
Operating Temperature Range	-40C to 85C
CPU	PIC32MX130F064B
CPU Clock Frequency	40Mhz
Analog Input A-to-D Resolution	10 bit
Firmware Version	2.5
DIN Voltage For Logic 0 (4)	<6.0V DC
DIN Voltage For Logic 1 (4)	>9.5V DC
Absolute Min/Max DIN Voltage (4)	+/- 30V
Typical DIN Current (DIN Volts)	2.3mA(12V), 5mA(24V)
Typical DIN Turn On/Off Time (5)	40/60 uS
Maximum DIN1 Pulse Frequency	10Khz
Opto-Iso DIN to DOUT Isolation	2500Vrms
Motor Current Per Phase Max	2.5A
Motor Step Frequency Max	25KHz

1. Typical current with no motor connected and no peripherals connected on the ML or MB ports.
2. The maximum current that can be supplied by the 3.3V voltage regulator on the ML I/O port.
3. Exceeding this voltage on the ML port I/O pin will damage the PCB.
4. Non-inverting DIN voltage values are referenced to C14. DIN "not connected" is guaranteed to return a logic 0.
5. Tested using a 12V square wave.

CAUTION

PCB IS SUSCEPTIBLE TO ELECTROSTATIC DAMAGE

CAUTION

EXCEEDING 3.3V ON I/O PINS WILL DAMAGE THE PCB

MADE IN USA

All solder, components and materials used to manufacture this product comply with the European Union RoHS (Reduction of Hazardous Substances) directive. Any company names, products or trademarks referenced are the property of their owners.

```
;=================================================================================
; Below are example motor control scripts. The example script must be loaded into
; array "MotorScript" in order to run. NOTE: Script1 is loaded by default.
;=================================================================================

;=================================================================================
; wait for PGM switch, move forward 3 secs, move backward 5 secs, start over
;=================================================================================
  Table Script1
;set the motor parameters
     WriteVar, 1000, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar,   On, MotorEnable
;---------------------------------------------------------------------------------
;wait for PGM switch
     Jump, 0, PGM, Off
;---------------------------------------------------------------------------------
;move motor fwd 3 secs
     WriteVar, 3000, MotorTime
     MoveTime, Fwd, 0, 0
;---------------------------------------------------------------------------------
;move motor bwd 5 secs
     WriteVar, 5000, MotorTime
     MoveTime, Bwd, 0, 0
;---------------------------------------------------------------------------------
;start over
     Jump, -18, 0, 0
  End Table
;=================================================================================
; change motor speed using DI2 (increase) and DI3 (decrease)
;=================================================================================
  Table Script2
;set the motor parameters
     WriteVar, 1000, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar,   50, MotorTime
     WriteVar,  100, Offset
     WriteVar,   On, MotorEnable
;---------------------------------------------------------------------------------
;change speed using DI2 (increase) and DI3 (decrease)
     Move, Fwd, 0b1100, Off
     Jump, 12, DI2, On
     Math, Subt, Offset, MotorSpeed
     Jump, 8, 0, 0
     Math, Add, Offset, MotorSpeed
;---------------------------------------------------------------------------------
;wait for both switches off with 50ms debounce time
     MoveTime, Fwd, 0, 0
     Jump, -4, 0b1100, On
     MoveTime, Fwd, 0, 0
     Jump, -32, 0, 0
  End Table
;=================================================================================
```

```
;================================================================================
; home the motor, move to set point, wait, move fwd/bwd 3 times, halt/blink red LED
;================================================================================
  Table Script3
;set the motor parameters
     WriteVar, 1000, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar, 7000, MotorSP
     WriteVar,   On, MotorEnable
;--------------------------------------------------------------------------------
;home the motor (motor home switch = DI1)
     Move, Bwd, DI1, Off
     WriteVar, 0, MotorPosn
;--------------------------------------------------------------------------------
;move to set point
     MoveSetPoint, 0, 0
;--------------------------------------------------------------------------------
;wait 2 secs
     WriteVar, 2000, WaitTime
     Wait
;--------------------------------------------------------------------------------
;loop 3 times
     WriteVar, 3, LoopCount
     WriteVar, 1000, Offset
     MoveOffset, Fwd, 0, 0
     Wait
     MoveOffset, Bwd, 0, 0
     Wait
     DecrJumpNZ, LoopCount, -10
;--------------------------------------------------------------------------------
;done, blink the red LED
     WriteVar, 1000, WaitTime
     WriteVar, 0b100, RGBLeds
     Wait
     WriteVar, 0b000, RGBLeds
     Wait
     Jump, -8, 0, 0
  End Table
;================================================================================
; wait for PGM switch, home the motor, continuously move to set point
;================================================================================
  Table Script4
;set the motor parameters
     WriteVar, 1000, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar, 7000, MotorSP
;--------------------------------------------------------------------------------
;wait for PGM switch
     Jump, 0, PGM, Off
;--------------------------------------------------------------------------------
;home the motor
     WriteVar, On, MotorEnable
     Move, Bwd, DI1, Off
     WriteVar, 0, MotorPosn
;--------------------------------------------------------------------------------
;continuously move to set point
     MoveSetPoint, 0, 0
     Jump, -3, 0, 0
  End Table
```

```
;=============================================================================
; wait for PGM switch, ramp speed to offset, move to SP, ramp down speed, start over
;=============================================================================
;                  _____________________
;  1000 (pps)    /|                     |\
;               / |                     | \
;              /  |                     |  \
;             /   |                     |   \
;            /    |                     |    \
;           /     |                     |     \
;          /      |                     |      \
;         /       |                     |       \
;  200   /        |                     |        \
;       0       5000                   8000      10000 (motor posn)

  Table Script5
;set the motor parameters
     WriteVar,  200, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar,   On, MotorEnable
;-----------------------------------------------------------------------------
;wait for PGM switch
     Jump, 0, PGM, Off
;-----------------------------------------------------------------------------
;ramp up the motor speed, stop at motor posn = 5000
     WriteVar,    0, MotorPosn
     WriteVar, 5000, Offset
     WriteVar,  200, Speed0
     WriteVar, 1000, Speed1
     MoveOffsetRamp, Fwd, 0, 0
;-----------------------------------------------------------------------------
;move to set point = 8000
     WriteVar, 8000, MotorSP
     MoveSetPoint, 0, 0
;-----------------------------------------------------------------------------
;ramp down the motor speed, stop at motor posn = 10000
     WriteVar, 2000, Offset
     WriteVar, 1000, Speed0
     WriteVar,  200, Speed1
     MoveOffsetRamp, Fwd, 0, 0
;-----------------------------------------------------------------------------
;start over
     Jump, -39, 0, 0
  End Table
;=============================================================================
; wait for PGM switch, ramp up speed over time, move for time, ramp down, start over
;=============================================================================
;                  _____________________
;  1000 (pps)    /|                     |\
;               / |                     | \
;              /  |                     |  \
;             /   |                     |   \
;            /    |                     |    \
;           /     |                     |     \
;          /      |                     |      \
;         /       |                     |       \
;  200   /        |                     |        \
;       0       4000                   7000      10000 (milliseconds)
```

```
  Table Script6
;set the motor parameters
     WriteVar,  200, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar,   On, MotorEnable
;-------------------------------------------------------------------------------
;wait for PGM switch
     Jump, 0, PGM, Off
;-------------------------------------------------------------------------------
;ramp up the motor speed over period of 4 seconds
     WriteVar, 4000, MotorTime
     WriteVar,  200, Speed0
     WriteVar, 1000, Speed1
     MoveTimeRamp, Fwd, 0, 0
;-------------------------------------------------------------------------------
;move for 3 seconds
     WriteVar, 3000, MotorTime
     MoveTime, Fwd, 0, 0
;-------------------------------------------------------------------------------
;ramp down the motor speed over period of 3 seconds
     WriteVar, 1000, Speed0
     WriteVar,  200, Speed1
     MoveTimeRamp, Fwd, 0, 0
;-------------------------------------------------------------------------------
;start over
     Jump, -34, 0, 0
  End Table
;===============================================================================
; wait for PGM switch, run the motor until DI1Counter >= Max, start over
;===============================================================================
  Table Script7
;set the motor parameters
     WriteVar, 500, MotorSpeed
     WriteVar, 500, MotorCurrent
     WriteVar,  On, MotorEnable
;-------------------------------------------------------------------------------
;wait for PGM switch
     Jump, 0, PGM, Off
;-------------------------------------------------------------------------------
;run motor one step at a time until DI1Counter >= Max
     WriteVar,   1, Offset
     WriteVar, 500, Max
     WriteVar,   0, DI1Counter
     MoveOffset, Fwd, 0, 0
     CompJump, -4, DI1Counter, LT, Max
;-------------------------------------------------------------------------------
;start over
     Jump, -22, 0, 0
  End Table
;===============================================================================
; wait for PGM switch, home the motor, read set point from 4/20mA input, move to SP
;===============================================================================
  Table Script8
;set the motor parameters
     WriteVar, 1000, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar,   On, MotorEnable
;-------------------------------------------------------------------------------
```

```
;wait for PGM switch
     Jump, 0, PGM, Off
;-------------------------------------------------------------------------------
;home the motor
     Move, Bwd, DI1, Off
     WriteVar, 0, MotorPosn
;-------------------------------------------------------------------------------
;read the set point from ML122 4/20mA input (address=0, channel=AI2)
     WriteVar,  4800, Adc0
     WriteVar, 24000, Adc1
     WriteVar,     0, Min
     WriteVar,  8500, Max
     ML122, 0, 2, Adc
     ReScale, Adc, MotorSP, Adc0, Min, Adc1, Max
;-------------------------------------------------------------------------------
;move to set point
     MoveSetPoint, 0, 0
     WriteVar, 100, WaitTime
     Wait
     Jump, -18, 0, 0
  End Table
;===============================================================================
; wait for DI1, turn on ML102 relay 1, wait for DI2, turn on relay 2, move fwd/back
;===============================================================================
  Table Script9
;set the motor parameters
     WriteVar, 1000, MotorSpeed
     WriteVar,  500, MotorCurrent
     WriteVar,   On, MotorEnable
     WriteVar, 2000, MotorTime
;-------------------------------------------------------------------------------
;wait for DI1
     Jump, 0, DI1, Off
;-------------------------------------------------------------------------------
;turn on ML102 relay 1 (address=0b100)
     WriteVar, b1, Data
     ML102, 0b100, Data
;-------------------------------------------------------------------------------
;wait for DI2
     Jump, 0, DI2, Off
;-------------------------------------------------------------------------------
;turn on ML102 relay 2 (address=0b100)
     WriteVar, b2, Data
     ML102, 0b100, Data
;-------------------------------------------------------------------------------
;wait for DI3
     Jump, 0, DI3, Off
;-------------------------------------------------------------------------------
;move forward 2 secs
     MoveTime, Fwd, 0, 0
;-------------------------------------------------------------------------------
;move backward
     Move, Bwd, 0, 0
  End Table
;===============================================================================
```