Reference Design

World of Motors

RD003601-1115

Overview

The World of Motors (WoM) Reference Design Board is a development platform built on Zilog's series of Mini-Z stamp modules. It is designed to provide engineers, students, and enthusiasts a simple-to-use platform for developing prototypes and projects related to motor control.

The WoM Board is designed to work with Zilog's series of Mini-Z modules. Additionally, it is compatible with the basic stamp modules available from multiple vendors. The WoM reference board includes an FTDI USB-to-serial converter for serial communication capability.

This reference design includes build options for using the Mini-Z shell or standalone operation. The Mini-Z shell provides a console through which the code is uploaded and executed via command line instructions. The standalone code requires the use of the ZDS tools to flash the Mini-Z (overwriting the shell code). The standalone version provides menu-driven commands to control the motors.

The World of Motors Reference Design Board is shown in Figure 1.

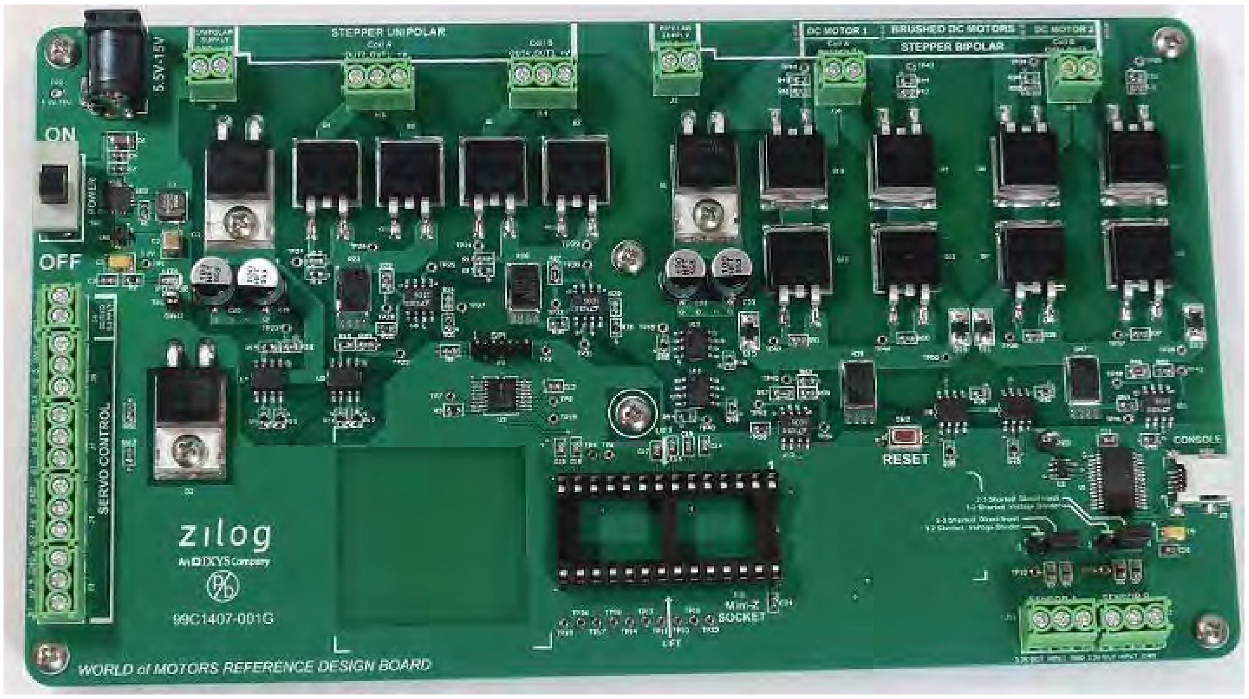


Figure 1. World of Motors Reference Design Board

Note: The source code file associated with this reference design, RD0036-SC01.zip, is available free for download from the Zilog website. The stand-alone version requires a USB Smart-Cable (not included in the World of Motors reference design) to program the ZNEO Mini-Z 28-pin module.

Features

The key features of this World of Motors reference design include:

- Controls the following motors:
  - DC brush motor
  - Servo motor
  - Unipolar stepper motor
  - Bipolar stepper motor
- Wirelessly controls motors using Mini-Z ZPAN or Mini-Z WLAN modules
- Provides console communication

Potential Applications

This reference design can be used to develop a number of applications; examples include:

- Robotics
- Security locks
- Fans
- Navigation systems
- Instrumentation
- Automated sprinkler systems

Hardware Design

The World of Motors Board is designed to control the following four kinds of motors:

Servo Motor. Servo motors are controlled using the General Purpose Input-Output (GPIO) timers of the ZNEO MCU.

DC Brush Motor. DC brush motors are controlled using PWMs through an H-Bridge circuit.

Unipolar Stepper Motor. Unipolar stepper motors are controlled with a MOSFET driver circuit.

Bipolar Stepper Motor. Bipolar stepper motors are controlled using an H-Bridge MOSFET driver circuit.

Figure 2 displays a block diagram of the World of Motors Board.

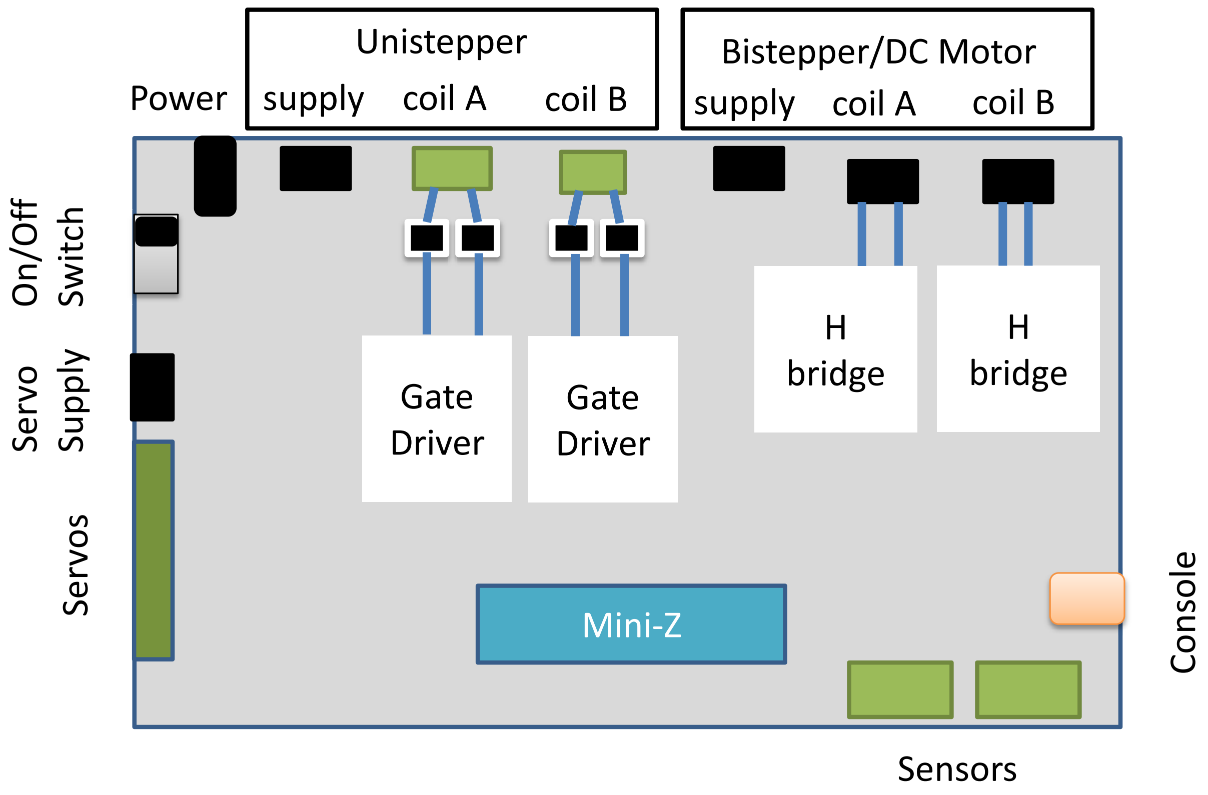


Figure 2. World of Motors Block Diagram

Theory of Operation

This section describes the operational principles of the World of Motors Reference Design.

MOSFET Driver

A MOSFET driver circuit is responsible for driving several motors used in this application. The MOSFET requires a gate drive voltage to turn ON, which in turn energizes the coils inside the motors, allowing them to run.

Figures 3 and 4 illustrate the ON and OFF state conditions of the MOSFET.

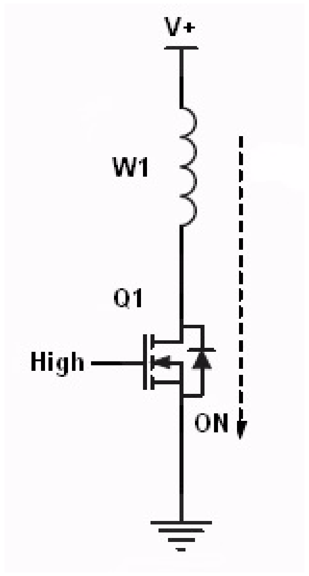


Figure 3. ON-State Condition of MOSFET

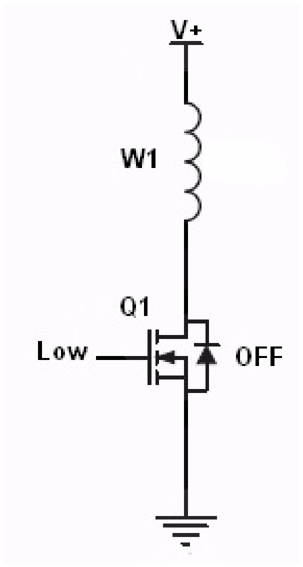


Figure 4. OFF-State Condition of MOSFET

H-Bridge Driver

An H-bridge circuit requires four MOSFETs configured in an H-letter pattern to control the motor. Figure 5 shows an H-bridge circuit.

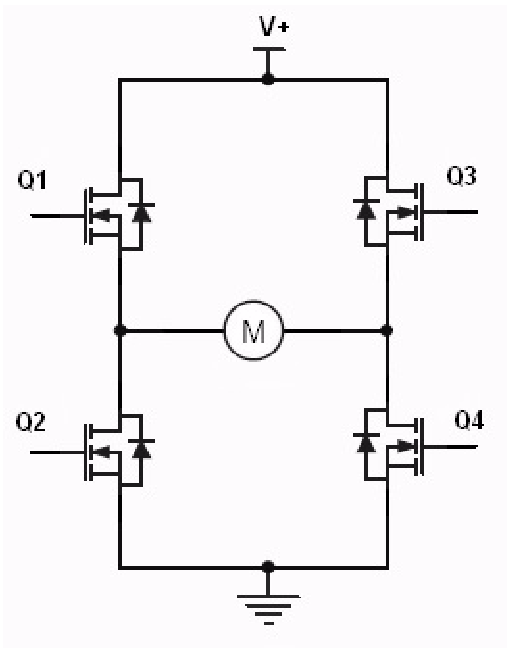


Figure 5. H-Bridge Circuit

Table 1 lists the different drive combinations for the H-bridge circuit to determine direction control.

Table 1. H-Bridge Circuit Drive Combinations

Direction	Q1	Q2	Q3	Q4
Clockwise	ON	OFF	OFF	ON
Counterclockwise	OFF	ON	ON	OFF
Free-run	OFF	OFF	OFF	OFF

Direction control is achieved by turning ON specific MOSFETs. For clockwise direction, transistor pair Q1-Q4 is turned ON, while Q2-Q3 is OFF. Figure 6 (a) illustrates the current flow in clockwise direction control.

For counterclockwise direction control, transistor pair Q2-Q3 is turned ON, while Q1-Q4 is OFF. Figure 6 (b) illustrates the current flow for counterclockwise direction. Figure 6 (c) illustrates a free-running condition in which all of the transistors are OFF. Trying to turn ON the MOSFETs on the same half-bridge on either side will cause a short circuit across the power supply.

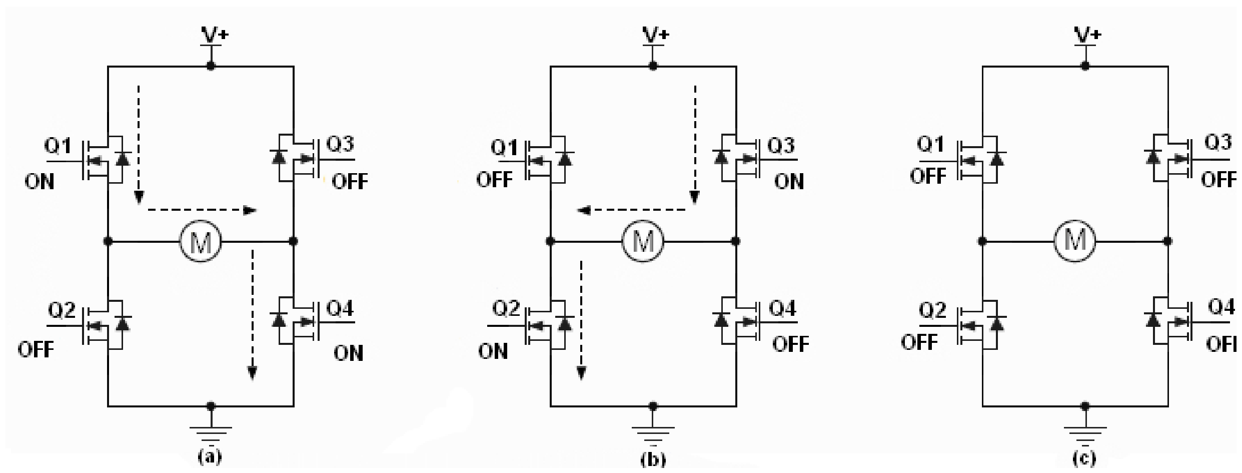


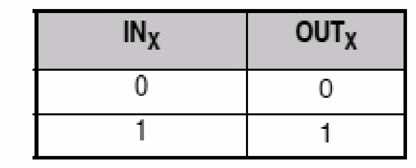
Figure 6. Current Flow in H-Bridge Drive Circuits

Speed control is achieved by replacing the steady state ON voltage with Pulse Width Modulation (PWM) pulses. For this reference design, PWM uses a fixed frequency with a variable duty cycle to control the speed. The average voltage applied to the motor is proportional to the PWM duty cycle. When the duty cycle is low, the speed of the motor is slow. When the duty cycle is high, the speed of the motor is fast. For example, in Figure 6 (a) and (b), the PWM pulse allows speed control in both directions.

MOSFET Gate Driver

The IXDN602 MOSFET gate driver from IXYS used in this reference design is a dual non-inverting driver. This gate driver is interfaced to every MOSFET and allows a high-current gate drive to the MOSFETs.

Figure 7 shows the truth table for the IXDN602 gate driver.



IN_X	OUT_X
0	0
1	1

Figure 7. MOSFET Gate Driver Truth Table

The H-bridge control uses P channel MOSFETs for the upper MOSFETS and N Channel MOSFETS for the lower ones to accommodate 5V rails without having to add a 12V rail and limit the gate voltage. To drive the P Channel and N channel gates, the IXDF602 Gate driver consists of one inverting and one non-inverting channel.

DC Motor

The DC motor consists of a stator, a rotor, and a metal frame to concentrate the flux. The attraction and repulsion of the poles causes the rotor to produce torque which makes it turn. When the rotor begins to turn, the fixed brush alternately achieves contact with the commutator. In this manner, the rotor windings are energized and de-energized as the rotor turns. The axes of the rotor poles are always opposite to the stator poles. If power supply to the motor terminals is reversed, the current flow in the rotor windings will also be reversed with respect to the magnet poles, resulting in a change in direction.

Figure 8 shows a DC brush motor.

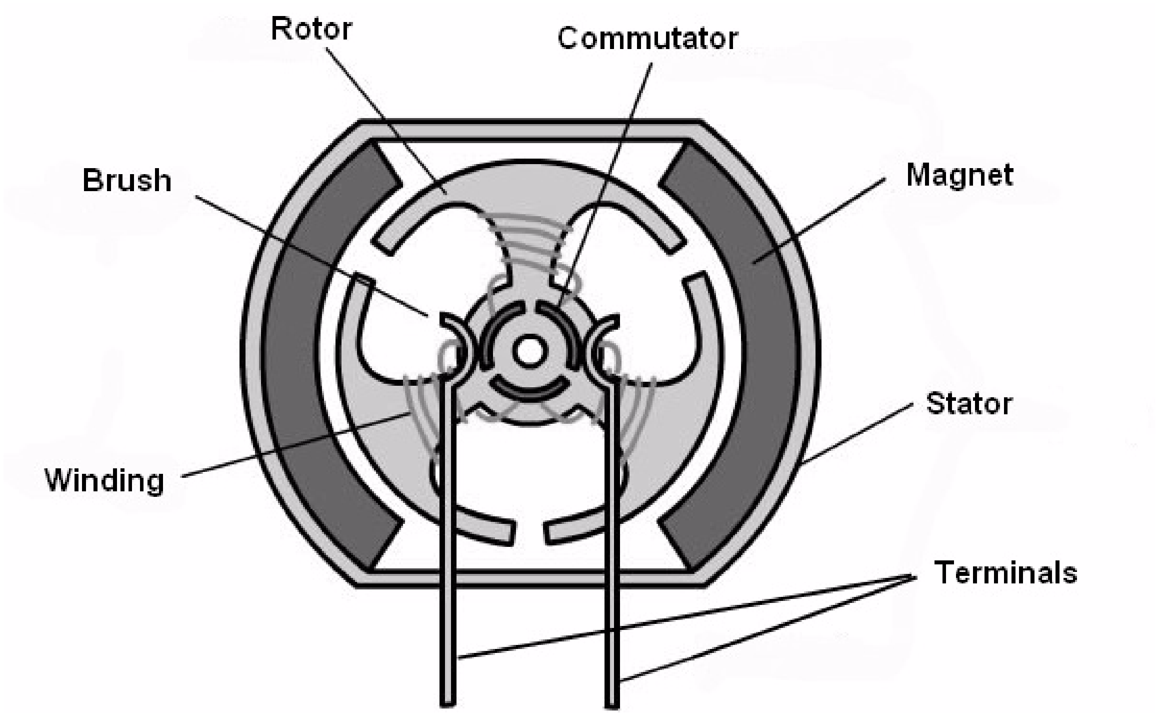


Figure 8. A Simple DC Brush Motor

Direction and speed control of the DC motor is implemented with the H-bridge circuit configuration. MOSFET pair Q1-Q4 is tied to PD0 (PWM1H pin), while Q2-Q3 is tied to PD1 (PWM1L pin). PD0 and PD1 pins provide a PWM pulse to control the DC motor. For clockwise direction, Q1-Q4 is injected with a PWM pulse, while Q2-Q3 is OFF. For counterclockwise direction, Q2-Q3 is injected with a PWM pulse, while Q1-Q4 is OFF. When all MOSFETs are turned OFF, the DC motor continues to free-run until it fully stops, regardless of load conditions. Speed control is managed by varying the duty cycle of the applied PWM pulse. The average voltage applied to the motor will be proportional to the PWM duty cycle. When the duty cycle is low, the speed of the motor is slow. When the duty cycle is high, the speed of the motor is fast. To start the DC motor, the duty cycle is set to a high value to overcome the initial startup current requirements and then decreased to the requested duty cycle.

Table 2 lists the different drive combinations for the H-bridge circuit to determine the direction and speed control.

Table 2. .H-Bridge Drive Combinations

Direction	Q1	Q2	Q3	Q4
Clockwise	PWM	OFF	OFF	PWM
Counterclockwise	OFF	PWM	PWM	OFF
Free-run	OFF	OFF	OFF	OFF

For example, in Figure 9, if the PWM duty cycle is set to 30%, the DC motor will run up to 30% of its full speed in a clockwise direction.

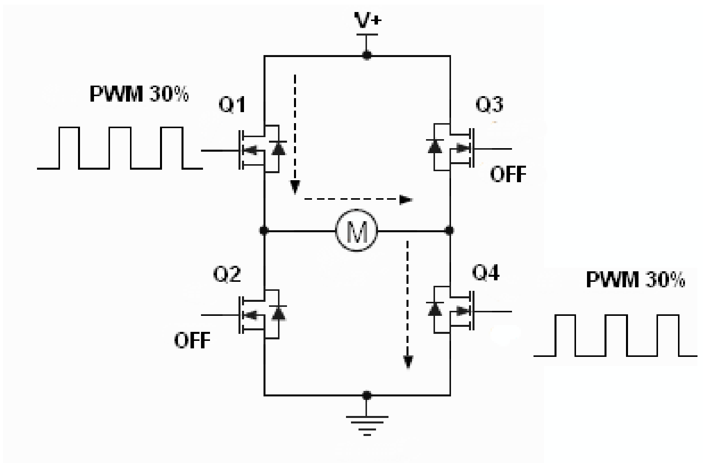


Figure 9. DC Motor Running at 30% Duty Cycle PWM in Clockwise Direction

In Figure 10, if the PWM duty cycle is set to 80%, the motor will run up to 80% of its full speed in counterclockwise direction.

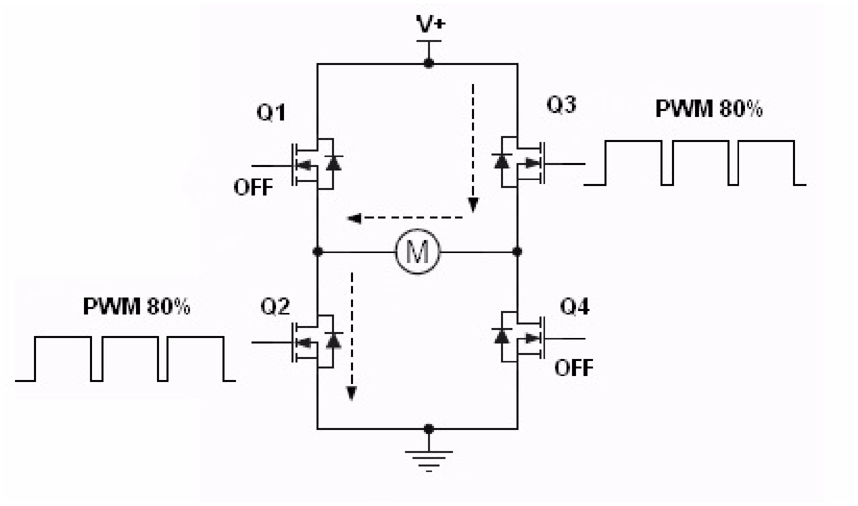


Figure 10. DC Motor Running at 80% Duty Cycle PWM in Counterclockwise Direction

Servo Motor

Servo motors are controlled using a timed pulse at varying lengths to control position and direction. Most servos run at a 20ms cycle with the ON pulse varying between 1.0ms and 2.0ms, where 1.5ms is the neutral point (0 degrees); this can vary depending on the servo. The constant rotation servos use the neutral point for the OFF position while the varied pulse determines speed and direction. Typically, the constant rotation servos use 1.3 ms for full speed in one direction while using 1.7ms for full speed in the other direction. Please see your specific servo motor's datasheet for actual values required.

Figure 11 shows the servo motor pulse in clockwise direction.

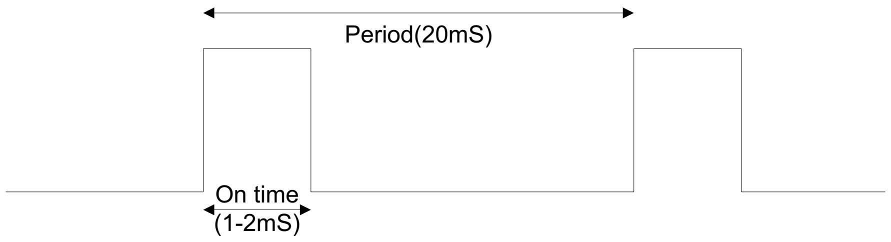


Figure 11. Servo Motor Pulse in Clockwise Direction

Stepper Motor

Stepper motors are characterized by high torque and are capable of handling large loads with precise movements. Figure 12 shows a typical stepper motor with four stator poles and two rotor poles. The stator poles are wound with a set of windings featuring two coils each, connected in series on opposite poles, while the rotor poles are permanent magnets comprised of soft alloy steel.

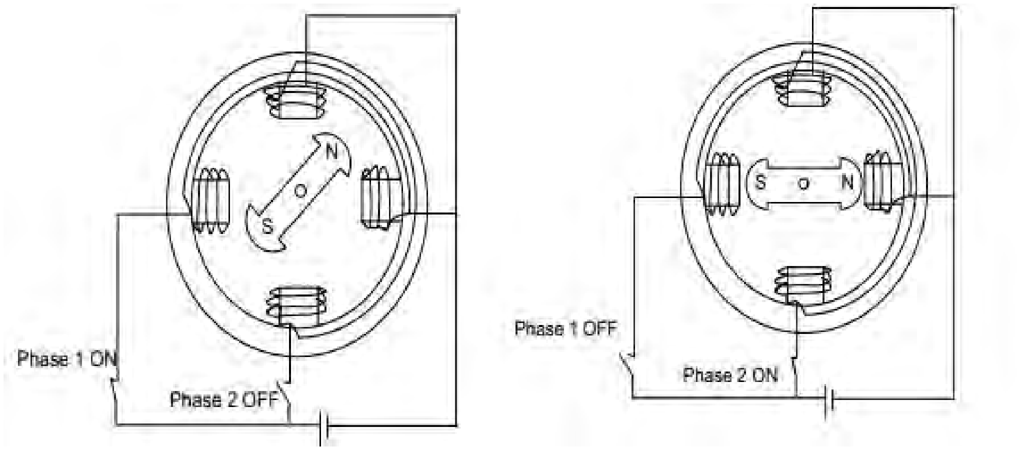


Figure 12. A Simple Stepper Motor

Sequentially energizing the two windings (Phase 1, Phase 2) causes the rotor poles to align with the electric poles and create rotor movement. The equilibrium states of the rotor are called *detent positions* and are fixed according to the mechanical structure of the motor. Stepper motors feature stator windings that are wound in unipolar or bipolar patterns on four or more poles.

This reference design allows for the use of two types of stepper motors – unipolar stepper motor and bipolar stepper motor. The MOSFET driver is used to control the unipolar stepper motor, while the 2 H-bridge circuit configuration is used to control the bipolar stepper motor.

Unipolar Stepper Motor

Four individual MOSFETs are used to energize the windings of the unipolar stepper motor. Specific current flows to the motor windings determine the rotor movement of the stepper motor. Figure 14 illustrates a simple unipolar stepper motor electrical connection.

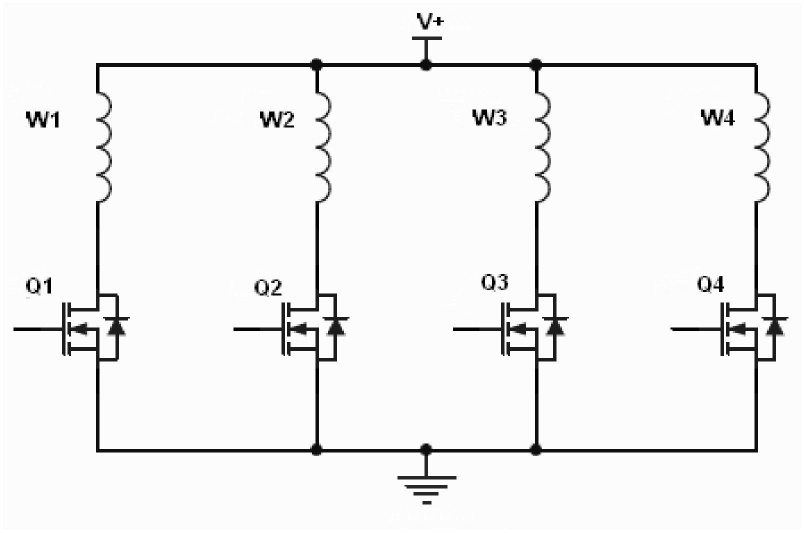


Figure 13. Unipolar Stepper Drive Scheme

Stepper motors can run in either full-step or half-step sequence. Tables 3 and 4 list the winding excitation steps for the unipolar stepper motor to run in either a full-step or a half-step sequence. To run the unipolar stepper motor in full-step seq in a clockwise direction, implement steps 1–4 in a repeating, ordered sequence as listed in Table 3. Full-step, counterclockwise direction is achieved by reversing the step sequence order.

Table 3. Unipolar Stepper Motor in Full Step

Step	Q1	Q2	Q3	Q4
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF

Table 4 lists the step sequences for half-step movement. To run the unipolar stepper motor in half-step sequence in a clockwise direction, implement steps 1 to 8 in a repeating, ordered sequence. Half-step, counterclockwise direction is achieved by reversing the step sequence order.

Table 4. Unipolar Stepper Motor in Half Step

Step	Q1	Q2	Q3	Q4
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	OFF
3	ON	OFF	OFF	ON
4	OFF	OFF	OFF	ON
5	OFF	ON	OFF	ON
6	OFF	ON	OFF	OFF
7	OFF	ON	ON	OFF
8	OFF	OFF	ON	OFF

Bipolar Stepper Motor

Two sets of H-bridge circuit configuration are used to energize the windings of the unipolar stepper motor, as shown in Figure 5. Motor movement is achieved by alternately reversing the current flow to the motor windings of the bipolar stepper motor. Similar to the unipolar stepper motor, the bipolar stepper motor can also move in full-step and half-step sequence.

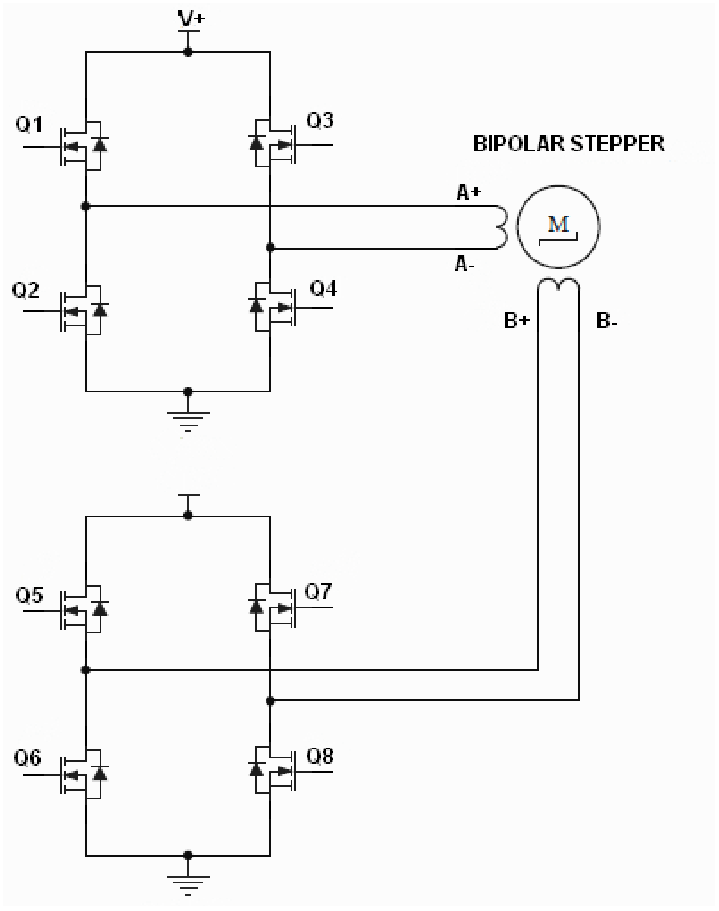


Figure 14. Bipolar Stepper Drive Scheme

Tables 5 and 6 list the winding excitation steps for the bipolar stepper motor to run in either full-step or a half-step sequence.

To run in full-step sequence in a clockwise direction, step 1 requires Q1-Q4 and Q5-Q8 to turn ON to energize the path from A+ to A- and B+ to B- resulting to a step forward in a clockwise direction. Step 2 requires Q1-Q4 and Q6-Q7 to turn ON to energize the path from A+ to A- and B- to B+ resulting in another step. Continuing with Steps 3 and 4 in ordered sequence will complete the full-step, clockwise direction cycle of the bipolar stepper motor. Repeating these steps over and over will allow for continuous rotation. Full-step, counterclockwise direction is implemented by reversing the order of sequence.

Table 5. Bipolar Stepper Motor in Full Step

Step	Q1-Q4	Q2-Q3	Q5-Q8	Q6-Q7
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF

Table 6 lists the steps for the bipolar stepper motor half-step movements. Clockwise and counterclockwise direction movement principles are similar to the full-step movement.

Table 6. Bipolar Stepper Motor in Half Step

Step	Q1-Q4	Q2-Q3	Q5-Q8	Q6-Q7
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	OFF
3	ON	OFF	OFF	ON
4	OFF	OFF	OFF	ON
5	OFF	ON	OFF	ON
6	OFF	ON	OFF	OFF
7	OFF	ON	ON	OFF
8	OFF	OFF	ON	OFF

Firmware

The WoM board allows for two different setups – one setup includes a bipolar stepper motor while the other setup connects two DC motors. Two different options for interfacing into the board are available:

– Mini-Z shell
– Standalone menu system

This reference design offers firmware with the following four build options:

- `WOM_Standalone_DCMotor` – Builds the project for the menu interface using the DC Motor code.
- `WOM_Standalone_BiPolar` – Builds the project for the menu interface using the Bipolar code.
- `WOM_Shell_DCMotor` – Builds the project for the Shell interface using the DC Motor code. The hex file is placed in the MiniZApp folder.
- `WOM_Shell_BiPolar` – Builds the project for the Shell interface using the Bipolar code. The hex file is placed in the MiniZApp folder.

Note: All four build options also support Unipolar and Servo motor control.

Standalone Interface

The standalone interface starts with the `main()` function in the `main.c` file. This function initializes the system clock and UART, and calls each motor's initialization routines, then transfers control to the `menu()` function located in the `menu.c` file. The `menu()` function presents the menu to users via the console (such as HyperTerminal or Tera Term) and allows the user to configure and control the different motors. The menu items call the appropriate API function to interact with the specific motor. When the application is built, download it via the USB SmartCable to the Mini-Z module being used.

Mini-Z Interface

The Mini-Z interface provides motor control functionality to the Mini-Z Shell. The Mini-Z interface starts with the `InitApplication()`function, which the shell calls when the `execapp` command is processed. The `InitApplication()` function registers the ports that will be controlled by this interface so that the shell commands will not use them. This function then calls each motor's initialization routines and registers the command handler with the shell. Control is sent back to the shell for normal operations. When a command comes into the shell through the console, the shell sends the command to the command handler that was registered (`WOMCMD_cmdProcessing()`). This function processes the request and calls the appropriate API to control the specific motor.

The World of Motors source code file, RD0036-SC01.zip, and the Mini-Z Library, RD0006-SC01, are available free for download from the Zilog website. To learn more about the library and functions, refer to the Mini-Z Shell and Flash Loader Reference Manual (RM0061).

To create a shell application, include the `miniz_startups.asm` file into your project and add the `Minizlib.lib` library file into your project settings. For more information, refer to the Mini-Z Shell and Flash loader Reference Manual (RM0061). Shell version 29 is used for this reference design.

Servo

The `servo.c` file contains the operations for the servos. The servos are connected to the PA6, PA7, PC4, and PC5 ports. The `SERVO_Init()` function initializes the ports and the SPI peripheral since PC4 and PC5 are shared with the servos. The servo uses a count-up Timer 1 for timing of the pulses to the servos. The period is set to 50 uS, to allow for 50 uS steps for servo positioning. The timer interrupt handles counting of the 50 uS periods to accommodate the proper pulse length and the 20 mS period.

The `SERVO_SetDuty()`function sets the duty cycle for the servo. Each step is 50 uS from the neutral position. The neutral position is set as a `SERVO_NEUTRAL` macro to provide a 1.5 mS pulse. If the servo is to be rotated in a clockwise manner, the step will be positive, and for a counter-clockwise rotation, the step will be negative. The servo maximum step value is set to ± 11, which gives the following result:

1.5mS + 11 * 50uS = 2.05mS, and

1.5mS - 11 * 50uS = 0.95mS

This allows a maximum rotation for most servos.

The interrupt service routine for the timer counts the number of 50uS periods and enables or disables the pulse signal, depending on the value of the duty cycle set for the different servos.

The PC4/PC5 ports are shared with the SPI via the MUX part. The MUX part uses two channels to connect PC4/PC5 to the servos or the SPI MOSI/MISO ports. SPI Signal Select and SPI clock pins are wired directly, so that the slave does not get confused when data stops coming in. The servo file also contains a few utilities to share the SPI, yet produce the servo signal on time. The `SERVO_SafeForSPI()` function is called whenever the SPI wants to send or receive a byte. This function ensures that there is sufficient time in the low state to be able to switch to SPI mode and switch back before it is time to send the next pulse. The `SERVO_SwitchPort()` function actually handles the switching of the ports by enabling the MUX channels and the alternative functions on PC4 and PC5. Because the SS pin is directly connected, the transaction can continue despite the possibility of a delay, while the pulse is sent to the servo, without confusing or stopping the transaction. There is no clock change as the clock is directly connected; therefore the SPI slave will not read the state of the MOSI/MISO pins when they are switched by the MUX.

Stepper Motor

The Stepper motor is controlled through a timer to set the period of movement between steps. The speed of the stepper motor cycling through the steps can be controlled by changing the period. The Unipolar motor uses Timer 0 whereas the Bipolar motor uses the I^{2}C timer.

Note: The Unipolar and Bipolar motors are both handled in the same manner; therefore, the generic term, *stepper motor control*, is used to refer to these motors. Any specific reference to either the Bipolar or Unipolar motor is clearly indicated. The function names are preceded by `XXXXX_` (`UNIPOLAR_` for unipolar motor and `BIPOLAR_` for bipolar motor).

Each step is defined in a 3-dimensional array that specifies the ports that should be ON and another array that controls the ports that should be OFF, depending on the direction. The steps are further divided by full step and half steps.

The stepper motor `XXXXX_Init()` function consists of setting the ports for output, enabling the timer and interrupts, and clearing the variables.

The `XXXXX_NextStep()` function does the work for stepping the motor. This function increments the index used for the third dimension of the arrays and clears the ports used for the stepper motor by “anding” the mask array, using the direction, type variables, and the index as the three indexes for the 3-dimensional array, then “or-ing” the ON array, again using the direction, type, and index as the indexes for the 3-dimensional array.

The `XXXXX_Step()` function handles the stepping for the stepper motor. The step value dictates the motor status (Off (0), single step (1), or continuous) and speed (2-100). If the step value is 0 or 1, disable the timer, as there is no additional step for which a timer is required. If the step value is greater than 1, set the timer period to the requested time and start the timer.

The `XXXXX_Direction()` function controls the direction by setting the direction variable as the index of the first dimension of the arrays.

The `XXXXX_StepType` function controls the type of step (half or full step) by setting the type variable as the index of the second dimension of the arrays.

The timer interrupt service routine calls the Next Step function to move the motor to the next step.

DC Motor

DC motors are controlled through the multi-channel PWM. Channel 1 controls DCMotor1 and Channel 2 controls DCMotor2. PWM is configured as independent, with the OUTCTL register controlling the output of the channels (high and low). This configuration provides the ability to turn off both the high and the low of the channel, or just one (which controls the direction). The speed is controlled by the duty cycle. Due to the nature of the coil, the duty cycle changes the “average” voltage to the coils by creating a low pass filter. As the “average” voltage changes, the speed changes.

The `DCMOTOR_ChgDirection()` function changes the direction of the motor. If the motor is running, stop the motor and change the direction. Restart the motor with the new direction.

The `DCMOTOR_SetSpeed()` function changes the duty cycle. If the motor is stopped, start the motor.

The `DCMOTOR_Stop()` function stops the motor by setting the bits of the OUTCTL register to disable the entire channel (both high and low).

The `DCMOTOR_Start()` function starts the motor. One of the inherent problems of all motors occurs during initial startup, when the coils take much more current to start the turning than is needed during operation. If a small duty cycle is applied to the DC motor, the motor may not turn because it may not have enough current to turn the motor to get it started. To accommodate this issue, when starting the motor, the `DCMOTOR_Start()` function is assigned a high duty cycle and activates the appropriate high or low part of the channel, depending on the current direction.

Note: Typically, the system monitors the current when starting. It waits for the current to drop, then decreases the duty cycle to hold the duty cycle at the requested speed (current is directly related to torque and back EMF is proportional to speed). This action requires advanced knowledge of the specifics of the motor. This feature has not been incorporated because the design is not motor-specific.

The PWM interrupt service routine handles the task of decreasing the duty cycle to the requested duty cycle during startup. The interrupt service routine's job is complete after the motor reaches the requested duty cycle.

Hardware Setup

The two hardware setups used to test this application are:

- Servo motor and DC motor
- Servo motor and stepper motors

The following motors are used for this reference design:

- Athlonix DC brush motor
- Parallax Continuous Rotation servo motor
- Portescap 55M048D1U unipolar stepper motor
- Portescap 35L048B1B bipolar stepper motor

Table 7 on page 24 lists the motor ratings.

Note: Refer to the documentation with each motor for specific wiring instructions. Direction conventions in this document may be reversed depending on how the motor is connected to the WoM board or which end of the motor is viewed.

Servo Motor and DC Motor Setup

Figure 15 shows the connection setup for the servo and DC motor combination.

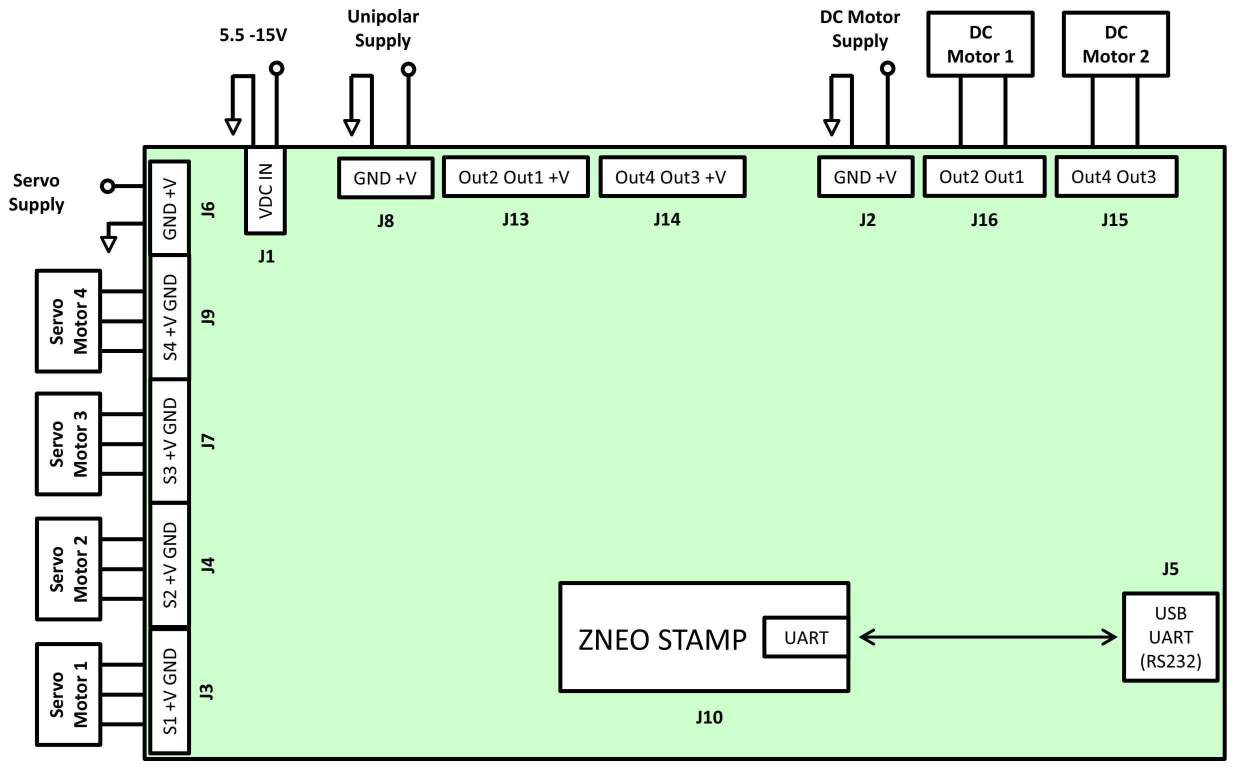


Figure 15. Servo and DC Motor Combination Connection Setup

Servo Motor, Unipolar Stepper Motor, and Bipolar Stepper Motor Setup

Figure 16 shows the connection setup for the servo and stepper motor combination.

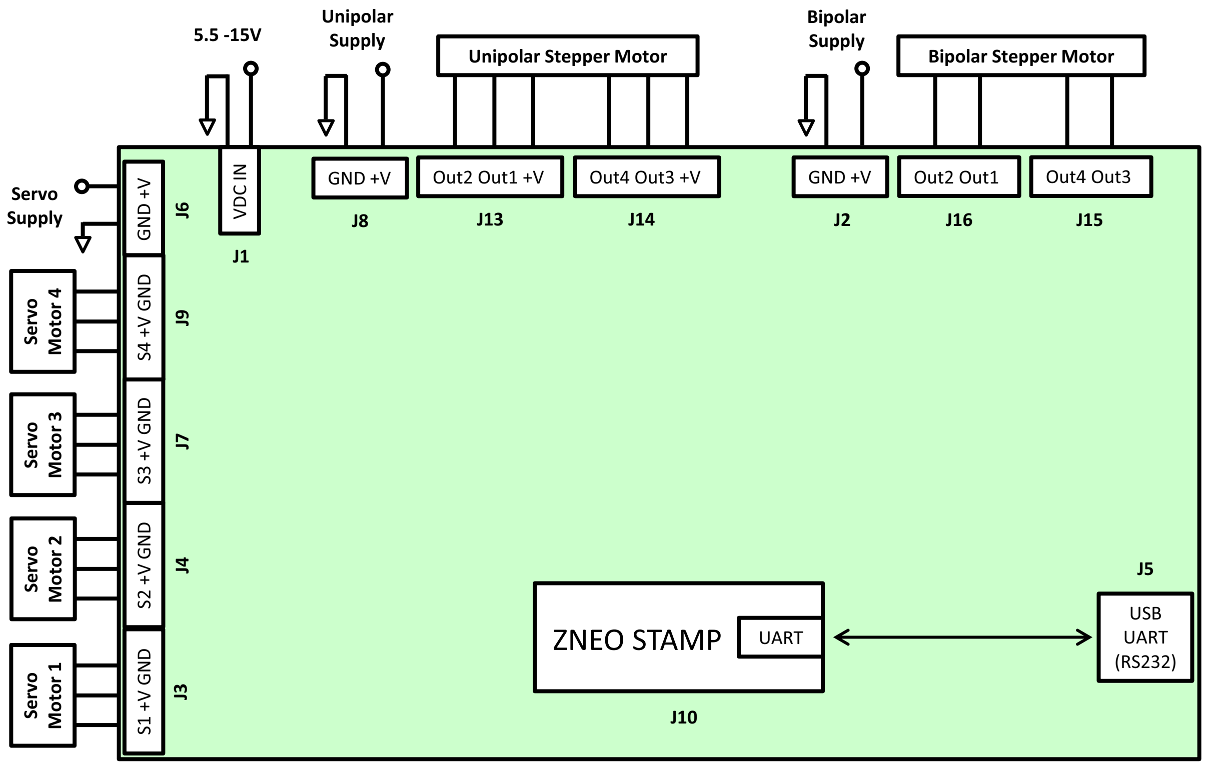


Figure 16. Servo and Stepper Motor Combination Connection Setup

Software Setup

Preparing the Software and Source Code

1. Download and install ZDS II ZNEO 5.2.0 Software and Documentation (Product ID SD00036) if you have not already done so. These files are available free for download from the Zilog Store.
2. Download RD0036-SC01.zip from the Zilog website and unzip it to the following location which is created during the installation process in Step 1.

```
< ZDSII_ZNEO_5.2.0 installation folder>\samples
```

Firmware Setup

Installing the Application

Perform the following steps to install the application:

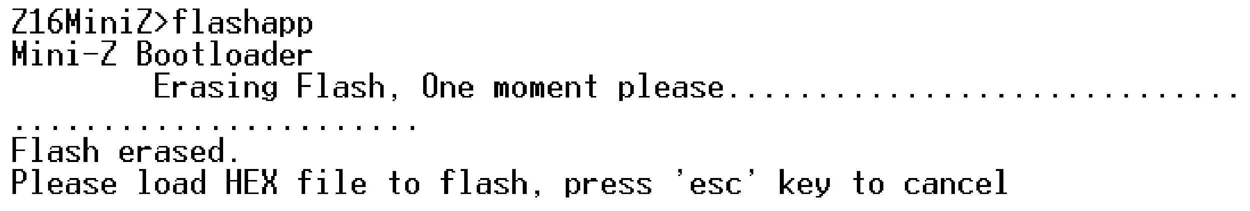
1. Run ZDSII ZNEO 5.2.0 and open the `wom.zdsproj` project located in the following directory:
   `< ZDSII ZNEO 5.2.0 installation folder>\samples\RD0036-SC01`
2. From the drop-down list, select the build option you wish to run.
3. From the ZDS II Menu, click **Rebuild All**.
4. Launch a terminal emulation program such as HyperTerminal or TeraTerm to establish a connection to the WoM Board. Configure the terminal application's communication setting with the following values:
   - 57600 baud rate
   - 8 data bits
   - Parity: none
   - Stop bit: 1
   - Flow control: none

Using the Shell Application

Observe the following procedure to use the Shell application:

Note: The Shell binaries are located in the Mini-Z Library/Bin folder. If the Mini-Z module has had the shell removed, use these hex files to load the shell into the module. Refer to the Mini-Z Shell and Flash loader Reference Manual (RM0061) for instructions.

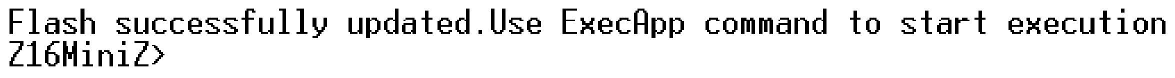
1. Enter the `FlashApp` command at the command prompt displayed by the shell (depending on the binary loaded in the Mini-Z module) when the WOM board is powered up, for example, `Z16MiniZ>.` The console displays a prompt to transfer a Hex file, as shown in Figure 17. From the **File** tab in the terminal program, select **Send Text File** to display the following dialog:



```
Z16MiniZ>flashapp
Mini-Z Bootloader
        Erasing Flash, One moment please.............................
.......................
Flash erased.
Please load HEX file to flash, press 'esc' key to cancel
```

Figure 17. Hex File Flashing Dialog

2. Navigate to the MiniZApp folder and select the `wom_dc_shell.hex` or `wom_stepper_shell.hex` file (depending the setup being used)
3. After the module Flash is successfully updated, the console returns to the command prompt, as shown in Figure 18.



```
Flash successfully updated.Use ExecApp command to start execution
Z16MiniZ>
```

Figure 18. Hex File Flash Result Dialog

4. Enter `Execapp` to run the WOM application.
5. Use the "?" command to see the new commands added for the World of Motors board. Type the command, followed by a space and the ? symbol, to see the help topic for the command.

Available Commands:

Servo	Controls the servos
Unisteptype	Configures the type of step (half step or full step)
Unistep	Controls the step, speed, and rotation direction

If the `WOM_DC_SHELL.`hex file is used:

DCMotor1	Controls DC Motor 1
DCMotor2	Controls DC Motor 2

If the `WOM_Stepper_Shell.hex` file is used:

Bisteptype	Configures the type of step (half step or full step)
BiStep	Controls the step, speed, and rotation direction

Note: Each time you power up the WoM Board, you must enter the `Execapp` command to execute the application.

Using the Standalone Menu Interface

Observe the following procedure to use the Standalone Menu interface:

1. Connect the USB SmartCable to the Mini-Z Module.
2. From the ZDS II menu, download the application to the Mini-Z Module.

3. Remove the SmartCable and press the reset switch on the WOM board. The terminal application should now display the menu to control the motors.

Main Menu:

If the `Standalone_DCMotor` build option is selected, the main menu features the following items:

1.	Servo Motors
2.	Unipolar Stepper Motor
3.	DC Motors
4.	Stop All Motors

When a build option other than `Standalone_DCMotor` is selected, the following options are presented:

1.	Servo Motors
2.	Unipolar Stepper Motor
3.	Bipolar Stepper Motor
4.	Stop All Motors

4. Select the motor you wish to control. Option 4 provides the ability to stop all motors and servos.

Servo Motors Menu:

C	Change Active Servo control
K	Disable Servo
F	Clockwise Direction
R	Counter Clockwise Direction
0-11	Position from Neutral (0 is Neutral)
B	Return to Main Menu

Option C changes the servo under menu control (1- 4). Option B returns to the main menu.

Unipolar and Bipolar Stepper Motors Menu:

F	Clockwise Direction
R	CounterClockwise Direction
H	Half Step operation

G	Full Step operation
0-100	Step (0-Disable;1-Single step;2-100;continous stepping)
B	Return to Main Menu

Option B returns to the main menu.

DC Motor Menu:

C	Change Active Motor control
S	Start Motor
K	Stop Motor
F	Clockwise Direction
R	CounterClockwise Direction
1-100	Speed
B	Return to Main Menu

Option C changes the DC motor under menu control (1 or 2). Option B returns to the main menu.

Electrical Specifications

Table 7 lists the electrical characteristics of the World of Motors Design Board and reflects all available data as a result of testing prior to qualification and characterization.

Table 7. Electrical Specifications for the World of Motors Design Board

Parameter	Minimum	Typical	Maximum	Units	Notes
V_{IN} range	5.5		17	Volts	
V_{SERVO}		5		Volts	
$V_{STEPPER}$ driver			35	Volts	
$V_{DCMOTOR}$ driver			35	Volts	

Note: The data presented in Table 7 is subject to change.

Packaging

Figure 19 displays an assembly diagram of the World of Motors Reference Design.

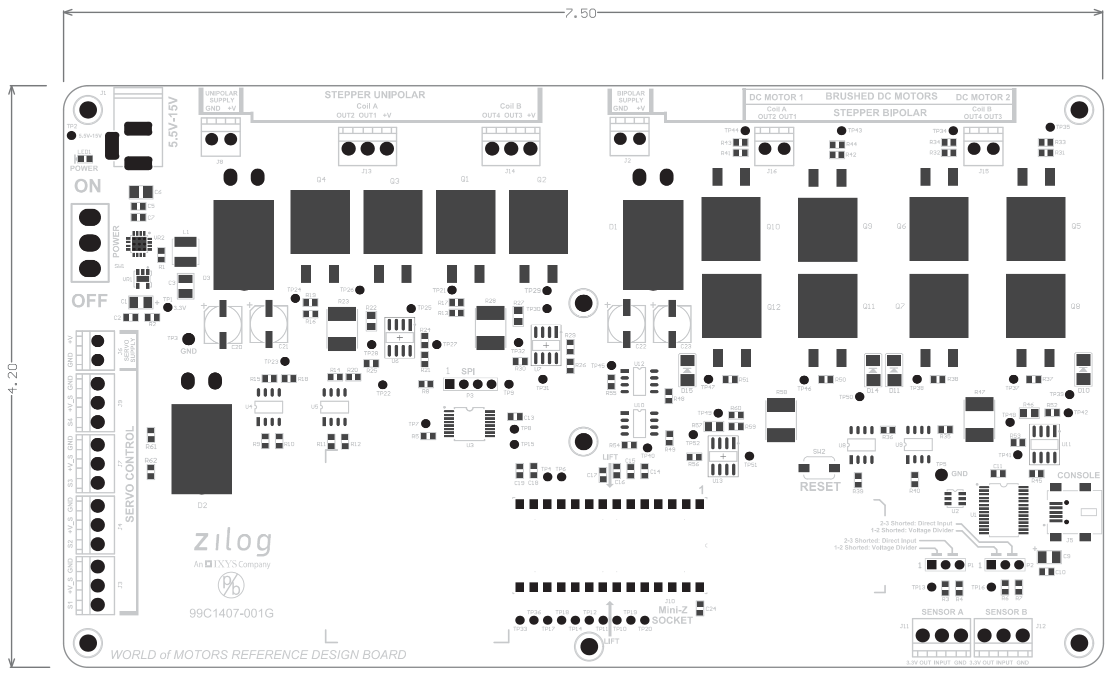


Figure 19. World of Motors Assembly Diagram

Ordering Information

The products associated with this World of Motors Reference Design are available individually or as a kit and can be ordered from the Zilog Store using the part numbers listed in Table 8.

Table 8. Ordering Information

Part Number	Description	Store Product ID
Z16F28WM100ZRDG	World of Motors Design Board	RD10017
Z16F2800100MODG	Mini-Z ZNEO 28-Pin Module	RD10002
Z16F28WF100MODG	Mini-Z WLAN 28-Pin Module	RD10003
Z16SSR00100KITG	Mini-Z ZPAN 28-Pin Module	RD10004

Kit Contents

The World of Motors Reference Design Kit contains the following items:

- Mini-Z ZNEO 28-Pin Module
- World of Motors Design Board
- Mini-Z to Standard debug adapter
- USB cable (A male to Mini-B male)
- DIP Package Extractor

Results

The firmware developed for the World of Motors Reference Design was successfully tested using the following motors:

- Servo motor (Parallax Continuous Rotation servo motor)
- DC motor (Athlonix DC brush motor)
- Unipolar stepper motor (Portescap 55M048D1U unipolar stepper motor)
- Bipolar stepper motor (Portescap 35L048B1B bipolar stepper motor)

Summary

The World of Motors Reference Design Board is a development platform built on Zilog's series of Mini-Z stamp modules. The WoM Board is designed to work with Mini-Z ZNEO, Mini-Z WLAN, and Mini-Z ZPAN. Additionally, it is compatible with the basic stamp modules available from other vendors.

This reference design includes shell and standalone application codes. The shell application is built upon an existing ZNEO Mini-Z library to provide command line instructions to drive the motors. The standalone application is a terminal-based application that provides menu-driven commands to drive the motors.

Related Documentation

The documents associated with the World of Motors Reference Design are listed in Table 9. Each of these documents can be obtained from the Zilog website by clicking the link associated with its document number.

Table 9. Related Documentation

Document Number	Description
RD0006	Mini-Z ZNEO 28-Pin Module Reference Design document
RD0006-SC01	Mini-Z Library
RM0061	Mini-Z Shell and Flash Loader Reference Manual
PS0220	ZNEO Z16F Series Product Specification
UM0188	ZNEO CPU Core User Manual
UM0181	USB Smart Cable User Manual

Appendix A. Flowcharts

Figure 20 displays a flow of the `SERVO_function` routine.

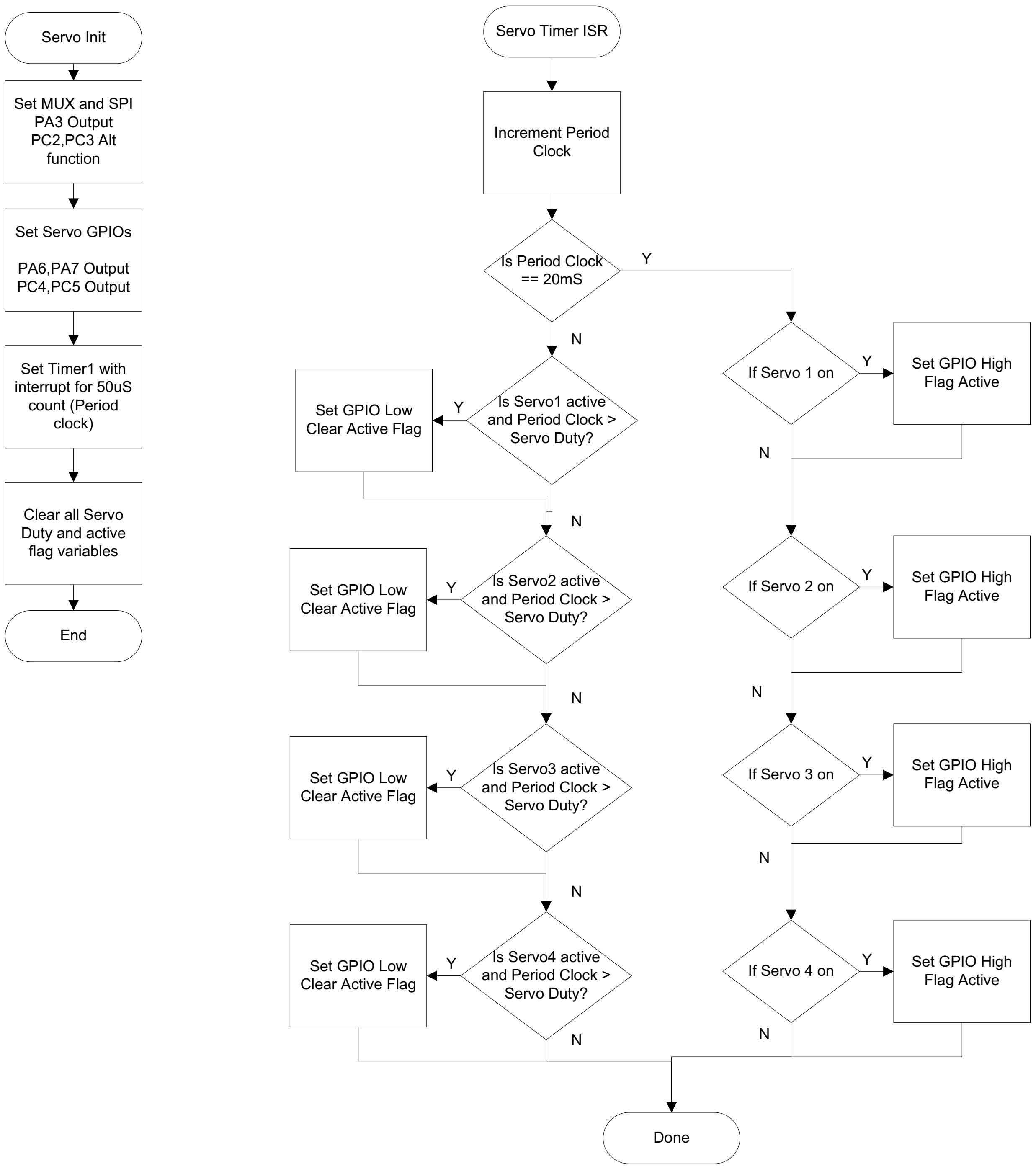

Servo Init
Set MUX and SPI PA3 Output PC2,PC3 Alt function
Set Servo GPIOs PA6,PA7 Output PC4,PC5 Output
Set Timer1 with interrupt for 50uS count (Period clock)
Clear all Servo Duty and active flag variables
End
Servo Timer ISR
Increment Period Clock
Is Period Clock == 20mS
Y
N
Is Servo1 active and Period Clock > Servo Duty?
Set GPIO Low Clear Active Flag
Is Servo2 active and Period Clock > Servo Duty?
Is Servo3 active and Period Clock > Servo Duty?
Is Servo4 active and Period Clock > Servo Duty?
If Servo 1 on
If Servo 2 on
If Servo 3 on
If Servo 4 on
Set GPIO High Flag Active
Done

Figure 20. Flowchart of the SERVO_function Routine

Figure 21 shows a flow of the `Unipolar_function` routine.

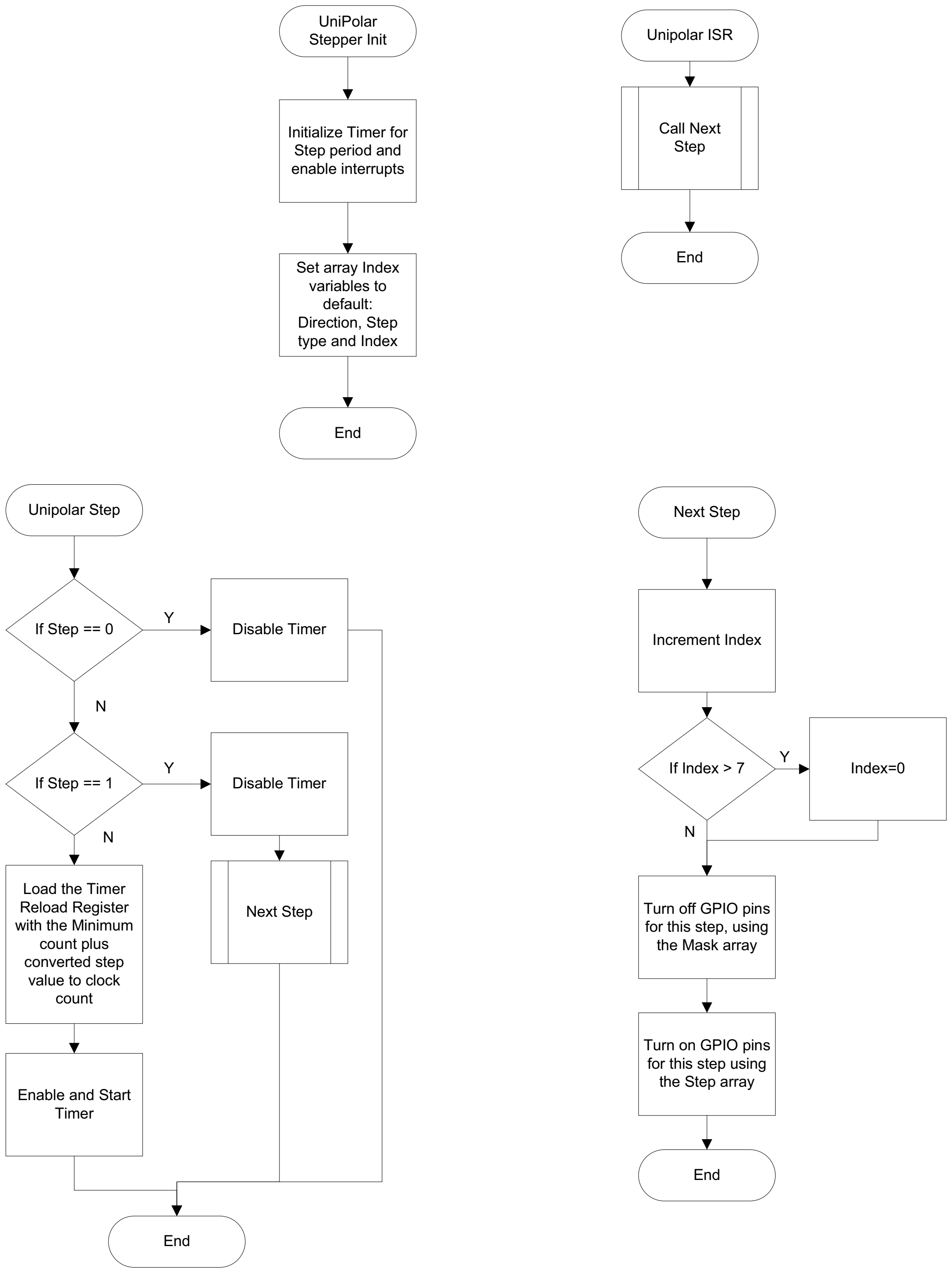

UniPolar Stepper Init
Initialize Timer for Step period and enable interrupts
Set array Index variables to default: Direction, Step type and Index
End
Unipolar ISR
Call Next Step
End
Unipolar Step
If Step == 0
Y
Disable Timer
N
If Step == 1
Y
Disable Timer
N
Load the Timer Reload Register with the Minimum count plus converted step value to clock count
Next Step
Enable and Start Timer
End
Next Step
Increment Index
If Index > 7
Y
Index=0
N
Turn off GPIO pins for this step, using the Mask array
Turn on GPIO pins for this step using the Step array
End

Figure 21. Flowchart of the Unipolar_function Routine

Figure 22 displays a flow of the `Bipolar_function` routine.

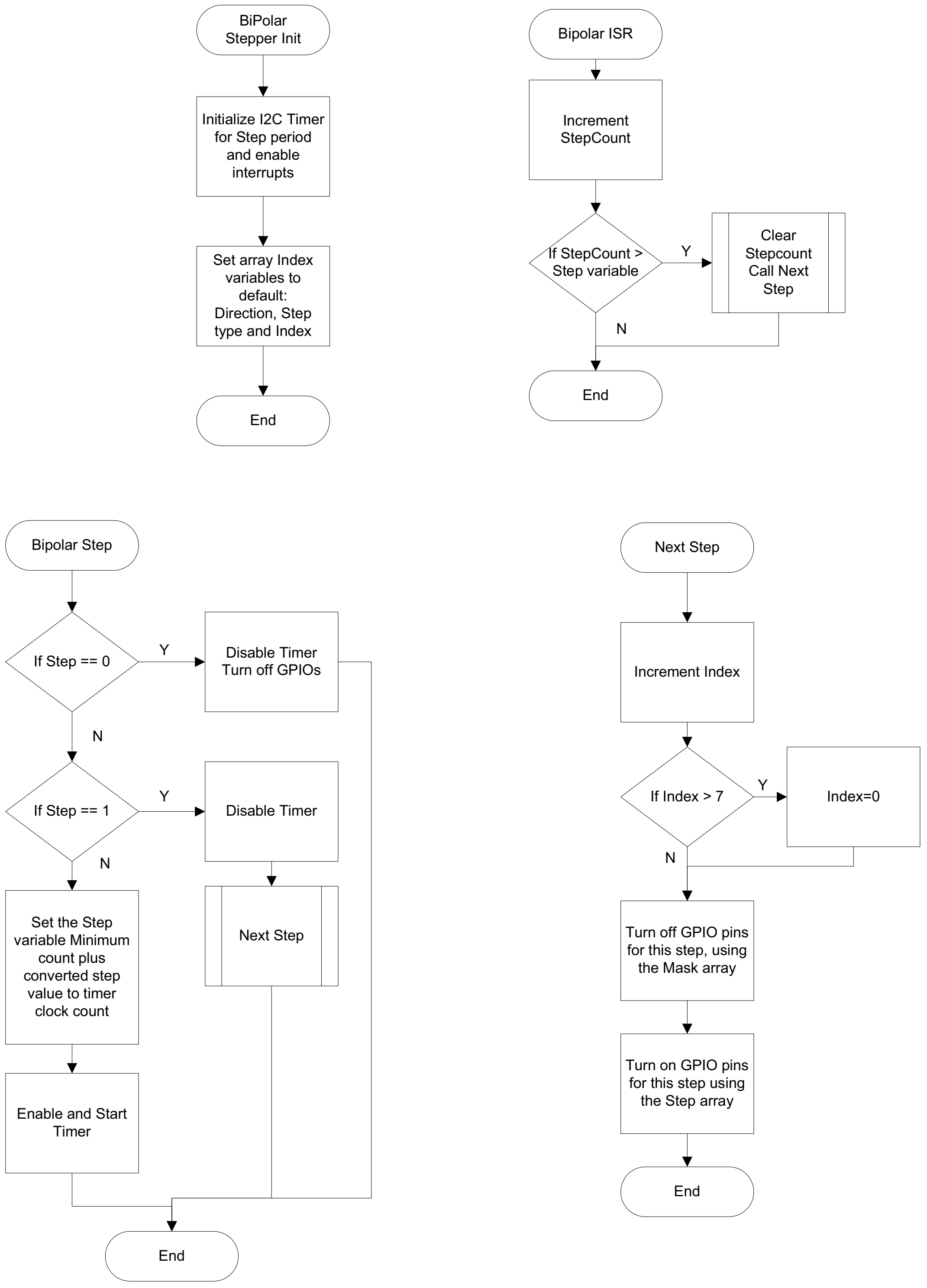

BiPolar Stepper Init
Initialize I2C Timer for Step period and enable interrupts
Set array Index variables to default: Direction, Step type and Index
End
Bipolar ISR
Increment StepCount
If StepCount > Step variable
Y
Clear Stepcount Call Next Step
N
End
Bipolar Step
If Step == 0
Y
Disable Timer Turn off GPIOs
N
If Step == 1
Y
Disable Timer
N
Set the Step variable Minimum count plus converted step value to timer clock count
Next Step
Enable and Start Timer
End
Next Step
Increment Index
If Index > 7
Y
Index=0
N
Turn off GPIO pins for this step, using the Mask array
Turn on GPIO pins for this step using the Step array
End

Figure 22. Flowchart of the Bipolar_function Routine

Figure 23 shows a flow of the `DCMOTOR_function` routine.

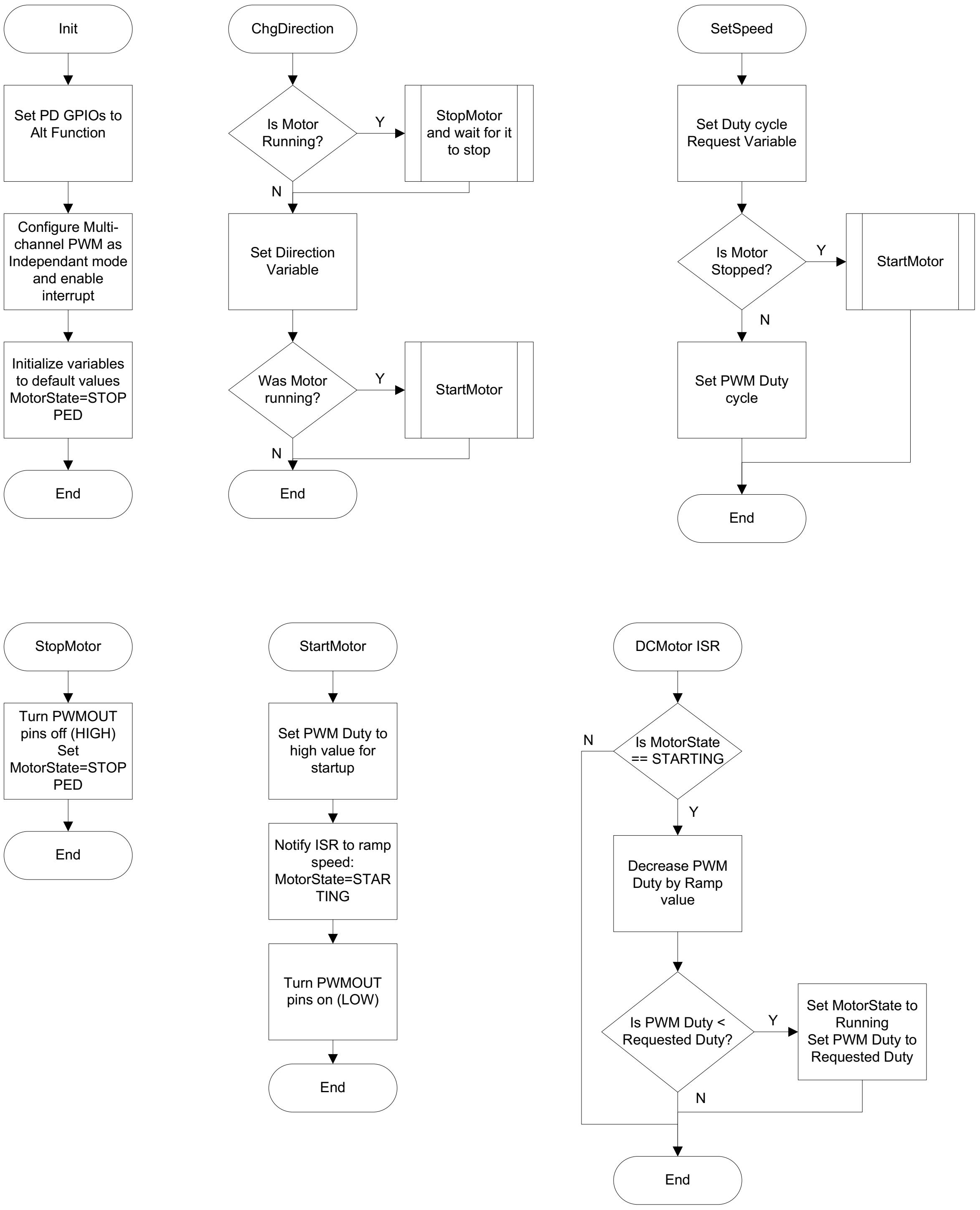

Init
Set PD GPIOs to Alt Function
Configure Multi-channel PWM as Independant mode and enable interrupt
Initialize variables to default values MotorState=STOPPED
End
ChgDirection
Is Motor Running?
Y
StopMotor and wait for it to stop
N
Set Diirection Variable
Was Motor running?
Y
StartMotor
N
End
SetSpeed
Set Duty cycle Request Variable
Is Motor Stopped?
Y
StartMotor
N
Set PWM Duty cycle
End
StopMotor
Turn PWMOUT pins off (HIGH) Set MotorState=STOPPED
End
StartMotor
Set PWM Duty to high value for startup
Notify ISR to ramp speed: MotorState=STARTING
Turn PWMOUT pins on (LOW)
End
DCMotor ISR
Is MotorState == STARTING
N
Y
Decrease PWM Duty by Ramp value
Is PWM Duty < Requested Duty?
Y
Set MotorState to Running Set PWM Duty to Requested Duty
N
End

Figure 23. Flowchart of the DCMOTOR_function Routine

Appendix B. Schematic Diagrams

Figures 24 through 26 show the schematic diagrams for the World of Motors Reference Design.

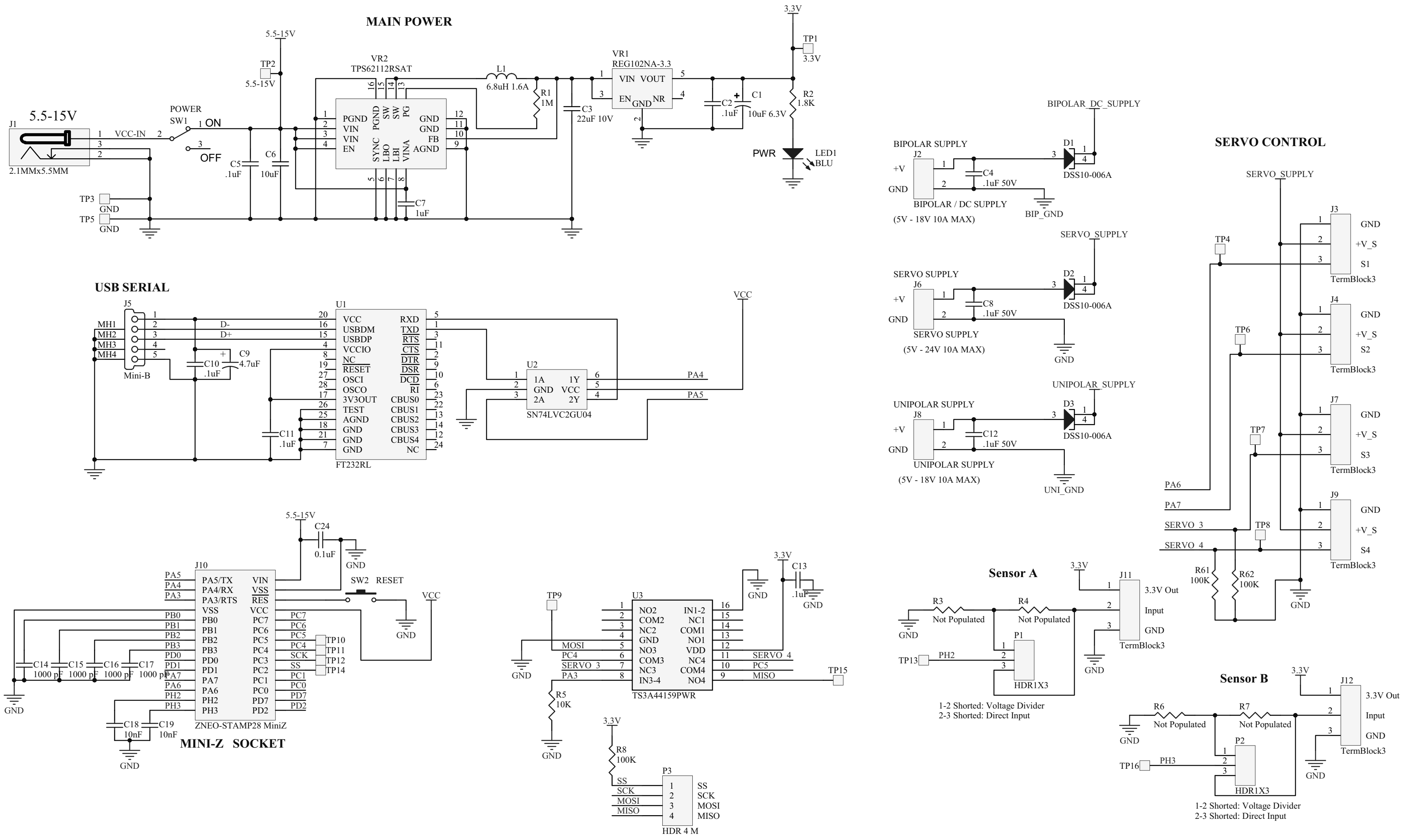


Figure 24. World of Motors Schematic Diagram, 1 of 3

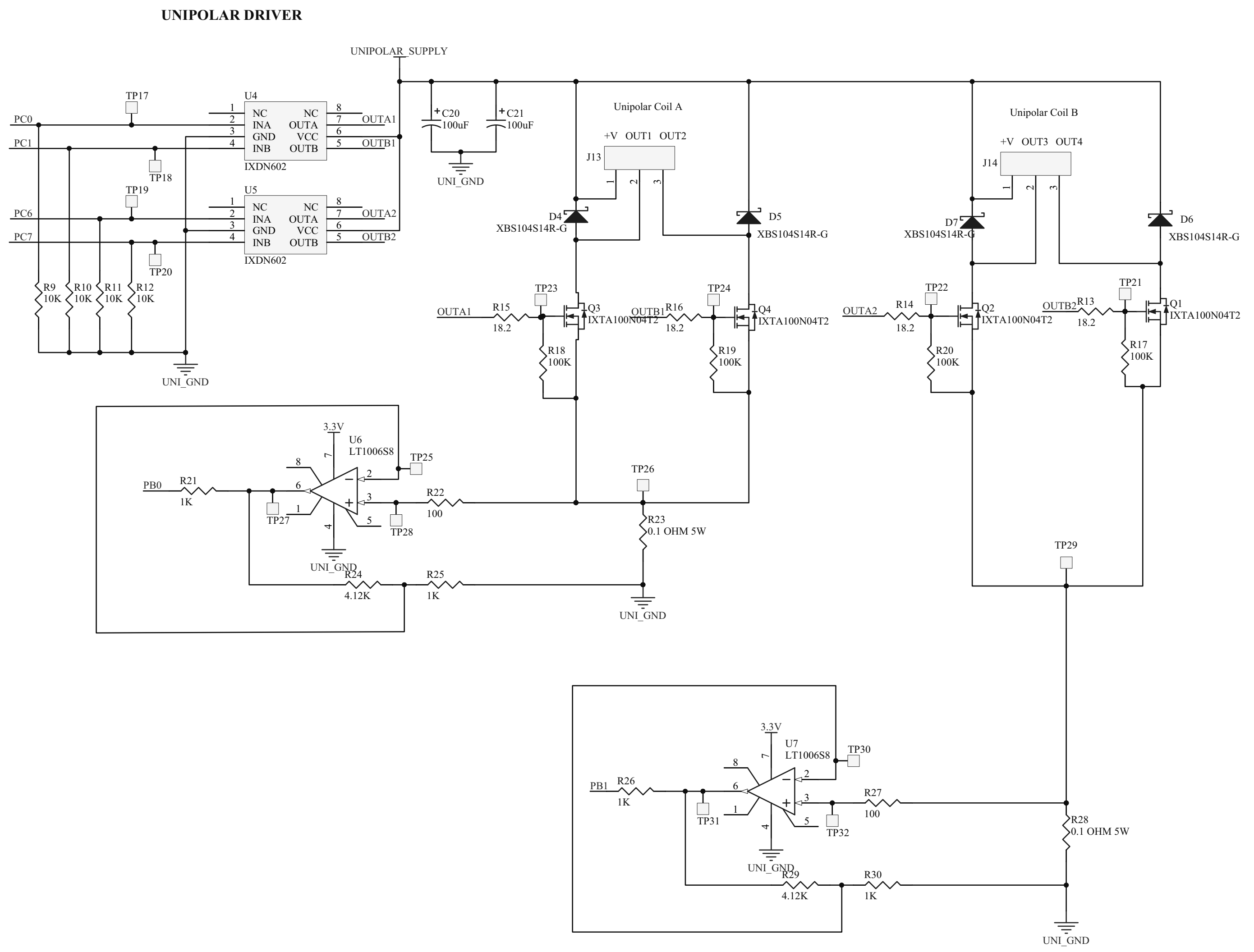


Figure 25. World of Motors Schematic Diagram, 2 of 3

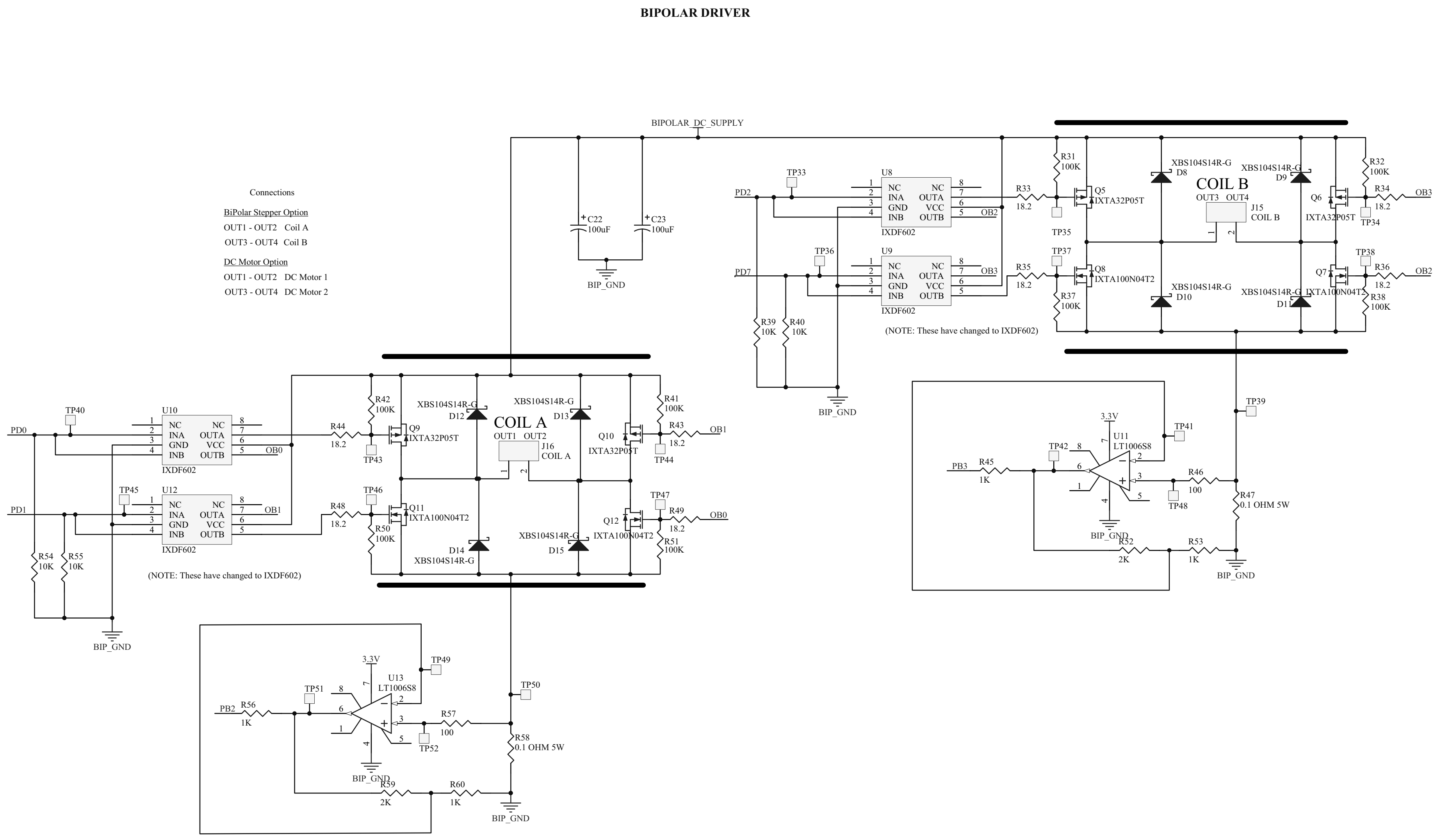


Figure 26. World of Motors Schematic Diagram, 3 of 3

Appendix C. Shell Commands

The World of Motors shell application provides commands to control the different motors. Tables 10 through 17 list the commands, their parameters and descriptions, and examples.

Table 10. Servo Command

Command	Servo
Description	This command controls the servo position. If this is a continuous rotation servo, the position number dictates the speed and direction of the rotation.
Parameters	Servo number (1-4) Servo Position (-11 to 11, 0 is neutral)
Example	Servo 1 11 Positions Servo 1 at the most clockwise position

Table 11. Unisteptype Command

Command	Unisteptype
Description	This command sets the step type for the stepper motor
Parameters	Step type either FULL or HALF
Example	Unisteptype FULL Sets the unipolar stepper motor to FULL step sequence

Table 12. Unistep Command

Command	Unistep
Description	This command sets the speed and/or the direction of the Unipolar Stepper motor
Parameters	Speed (0-100, where 0 is off, 1 is a single step) Direction (optional: CW for clockwise, CCW for counter clockwise)
Example	Unistep 1 CCW Steps the Unipolar stepper motor 1 step in the counter clockwise position. If the direction is not included, the step continues in the last direction given

Table 13. Bisteptype Command

Command	Bisteptype
Description	This command sets the step type for the stepper motor

Table 13. Bisteptype Command

Parameters	Step type either FULL or HALF
Example	Bisteptype FULL Sets the Bipolar stepper motor to FULL step sequence.

Table 14. Bistep Command

Command	Bistep
Description	This command sets the speed and/or the direction of the Bipolar Stepper motor
Parameters	Speed (0-100, where 0 is off, 1 is a single step) Direction (optional: CW for clockwise, CCW for counter clockwise
Example	Bistep 1 CCW Steps the Bipolar stepper motor 1 step in the Counter clockwise position. If the direction is not included, the step continues in the last direction given "\r\n\r\nSets speed and optionally the direction of the DCMOTOR1. \r\nCmd: \r\nDCMOTOR1 [0-100] (0 turns off the motor) [CW/CCW]\r\n", "\r\n\r\nSets speed and optionally the direction of the DCMOTOR2. \r\nCmd: \r\nDCMOTOR2 [0-100] (0 turns off the motor) [CW/CCW]\r\n",

Table 15. DCMotor1 Command

Command	DCMotor1
Description	This command sets the speed and/or the direction for the DC Motor 1
Parameters	Speed (0-100, 0 turns off the motor) Direction (optional: CW for clockwise, CCW for counter clockwise)
Example	DCMotor1 100 CW DCMotor1 will run at full speed in a clockwise direction

Table 16. DCMotor2 Command

Command	DCMotor2
Description	This command sets the speed and/or the direction for the DC Motor 2

Table 16. DCMotor2 Command

Parameters	Speed (0-100, 0 turns off the motor) Direction (optional: CW for clockwise, CCW for counter clockwise)
Example	DCMotor2 100 CW DCMotor2 will run at full speed in a clockwise direction

Table 17. StopMotors Command

Command	StopMotors
Description	This command stops all running motors and servos
Parameters	None
Example	Stopmotors Stops all running motors and servos

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.