## TETRIX PRIZM Robotics Controller Arduino Library:

Below is a reference of each function statement in the TETRIX PRIZM Robotics Controller Library for Arduino. For a detailed description of each function statement, please download the full *TETRIX PRIZM Programming Guide* found at **www.TETRIXrobotics.com/PRIZMdownloads**.

**PrizmBegin();**
**PrizmEnd();**
**setGreenLED(HIGH/LOW);**
**setRedLED(HIGH/LOW);**
**setMotorPower(motor#, power);**
**setMotorPowers(power1, power2);**
**setMotorSpeed(motor#, speed);**
**setMotorSpeeds(speed1, speed2);**
**setMotorTarget(motor#, speed, target);**
**setMotorTargets(speed1, target1, speed2, target2);**
**setMotorDegree(motor#, speed, degrees);**
**setMotorDegrees(speed1, degrees1, speed2, degrees2);**
**setMotorInvert(motor#, invert);**
**readMotorBusy(motor#);**
**readMotorCurrent(motor#);**
**readEncoderCount(enc#);**
**readEncoderDegrees(enc#);**
**resetEncoder(enc#);**
**readLineSensor(port#);**
**readSonicSensorCM(port#);**
**readSonicSensorIN(port#);**
**readBatteryVoltage();**
**readStartButton();**
**setServoSpeed(servo#, speed);**
**setServoSpeeds(speed1, speed2, speed3, speed4, speed5, speed6);**
**setServoPosition(servo#, position);**
**setServoPositions(position1, position2, position3, position4, position5, position6);**
**readServoPosition(servo#);**
**setCRServoState(CRservo#, state);**

## Appendix A: TETRIX PRIZM Library functions

| | | |
|---|---|---|
| **Prizm Begin**<br>Called in the Arduino code setup() loop. Initializes the PRIZM Controller. | PrizmBegin(); | PrizmBegin();<br>*Reset and initialize PRIZM controller.* |
| **Prizm End**<br>When called, immediately terminates a program and resets the PRIZM controller. | PrizmEnd(); | PrizmEnd();<br>*Terminate a PRIZM program and reset controller.* |
| **Set Red LED**<br>Sets the state of the PRIZM red indicator LED to On or Off. | setRedLED(); | setRedLED(HIGH);<br>*Turn red LED on.*<br>setRedLED(LOW);<br>*Turn red LED off.* |
| **Set Green LED**<br>Sets the state of the PRIZM green indicator LED to On or Off. | setGreenLED(); | setGreenLED(HIGH);<br>*Turn green LED on.*<br>setGreenLED(LOW);<br>*Turn green LED off.* |
| **Set DC Motor Power**<br>Sets the power level and direction of a TETRIX DC motor connected to the PRIZM DC motor ports. Power level range is 0 to 100. Direction is set by the (+/-) sign of the power level. Power level 0 = stop in coast mode. Power level 125 = stop in brake mode. | setMotorPower(); | setMotorPower(1, 50);<br>*Spin Motor 1 clockwise at 50% power.*<br>setMotorPower(2, -50%);<br>*Spin Motor 2 counter-clockwise at 50% power.*<br>setMotorPower(1, 0);<br>*Turn off Motor 1 in coast mode.*<br>setMotorPower(2, 125);<br>*Turn off Motor 2 in brake mode.* |
| **Set DC Motor Powers**<br>Simultaneously sets the power level and direction of ***both*** TETRIX DC motors connected to the PRIZM motor ports. Both PRIZM Motor 1 and Motor 2 channel parameters are set with a single statement. The Power level range is 0 to 100. Direction is set by the (+/-) sign of the power level. Power level 0 = stop in coast mode. Power level 125 = stop in brake mode. | setMotorPowers(); | setMotorPowers(50, 50);<br>*Spin Motor 1 and Motor 2 clockwise at 50% power.*<br>setMotorPowers(-50, 50);<br>*Spin Motor 1 counter-clockwise and Motor 2 clockwise at 50% power.*<br>setMotorPowers(0, 0);<br>*Turn off Motor 1 and Motor 2 in coast mode.*<br>setMotorPowers(125, 125);<br>*Turn off Motor 1 and Motor 2 in brake mode.* |
| **Set DC Motor Speed**<br>Uses velocity PID control to set the constant speed rate of a TETRIX DC motor with a TETRIX motor encoder installed. The speed parameter range is 0 to 720 degrees per second (DPS). The (+/-) sign of the speed parameter controls direction of rotation. | setMotorSpeed(); | setMotorSpeed(1, 360);<br>*Spin Motor 1 clockwise at a constant speed of 360 DPS.*<br>setMotorSpeed(1, -360);<br>*Spin Motor 1 counter-clockwise at a constant speed of 360 DPS.* |

| | | |
|---|---|---|
| **Set DC Motor Speeds**<br>Uses velocity PID control to simultaneously set the constant speed rate of ***both*** TETRIX DC motor channels with a TETRIX motor encoder installed. Both PRIZM Motor 1 and Motor 2 channel parameters are set with a single statement. The speed parameter range is 0 to 720 degrees per second (DPS). The (+/-) sign of the speed parameter controls direction of rotation. | setMotorSpeeds(); | setMotorSpeed(360, 360);<br>*Spin Motor 1 and Motor 2 clockwise at a constant speed of 360 DPS.*<br>setMotorSpeed(360, -360);<br>*Spin Motor 1 clockwise and Motor 2 counter-clockwise at a constant speed of 360 DPS.*<br>setMotorSpeed(360, -180);<br>*Spin Motor 1 clockwise and Motor 2 counter-clockwise at a constant speed of 180 DPS.* |
| **Set DC Motor Target**<br>Implements velocity and positional PID control to set the constant speed rate and the encoder count target holding position of a TETRIX DC motor with a TETRIX encoder installed. The speed parameter range is 0 to 720 degrees per second (DPS). The encoder count target position is a signed long integer from -2,000,000,000 to 2,000,000,000. Each encoder count = ¼ degree resolution. | setMotorTarget(); | setMotorTarget(1, 360, 1440);<br>*Spin Motor 1 at a constant speed of 360 DPS until encoder 1 count equals 1440. Once at encoder target count, hold position in a servo like mode.*<br>setMotorTarget(2, 180, -1440);<br>*Spin Motor 1 at a constant speed of 360 DPS until encoder 1 count equals -1440 (1 revolution). Once at encoder target count, hold position in a servo like mode.* |
| **Set DC Motor Targets**<br>Implements velocity and positional PID control to simultaneously set the constant speed rate and the encoder count target holding positions of ***both*** TETRIX DC motor channels each with TETRIX encoders installed. Both PRIZM Motor 1 and Motor 2 channel parameters are set with a single statement. The speed parameter range is 0 to 720 degrees per second (DPS). The encoder count target position is a signed long integer from -2,000,000,000 to 2,000,000,000. Each encoder count = ¼ degree resolution. | setMotorTargets(); | setMotorTargets(360, 1440, 360, 1440);<br>*Spin Motor 1 and Motor 2 at a constant speed of 360 DPS until each motor encoder count equals 1440. Once a motor reaches its encoder target count, hold position in a servo like mode.*<br>setMotorTargets(360, 1440, 180, 2880);<br>*Spin Motor 1 at a constant speed of 360 DPS until encoder 1 count equals 1440. Spin Motor 2 at a constant speed of 180 DPS until encoder 2 equals 2880. Each motor will hold its position in a servo like mode once it reaches the encoder target.*<br><br>*Note: 1 encoder count equal ¼ degree resolution. For example 1 motor revolution equals 1440 encoder counts (1440 / 4 = 360).* |

| | | |
|---|---|---|
| **Set Motor Degree**<br>Implements velocity and positional PID control to set the constant speed rate and the degrees target holding position of a TETRIX DC motor with a TETRIX encoder installed. The speed parameter range is 0 to 720 degrees per second (DPS). The encoder degrees target position is a signed long integer from -2,000,000,000 to 2,000,000,000 with a 1 degree resolution. | setMotorDegree(); | setMotorDegree(1, 180, 360);<br>*Spin Motor 1 at a constant speed of 180 DPS until encoder 1 degree count equals 360. Once at encoder target degree count, hold position in a servo like mode.*<br>setMotorDegree(2, 90, 180);<br>*Spin Motor 2 at a constant speed of 90 DPS until encoder 1 degree count equals 180. Once at encoder target degree count, hold position in a servo like mode.* |
| **Set Motor Degrees**<br>Implements velocity and positional PID control to set the constant speed rate and the degrees target holding position of ***both*** TETRIX DC motor channels with a TETRIX encoder installed. Both PRIZM Motor 1 and Motor 2 channel parameters are set with a single statement. The speed parameter range is 0 to 720 degrees per second (DPS). The encoder degrees target position is a signed long integer from -2,000,000,000 to 2,000,000,000 with a 1 degree resolution. | setMotorDegrees(); | setMotorDegrees(180, 360, 180, 360);<br>*Spin Motor 1 and Motor 2 at a constant speed of 180 DPS until each motor encoder degree count equals 360. Once a motor reaches its degree target count, hold position in a servo like mode.*<br>setMotorDegrees(360, 720, 90, 360);<br>*Spin Motor 1 at a constant speed of 360 DPS until encoder 1 degree count equals 720. Spin Motor 2 at a constant speed of 90 DPS until encoder 2 degree equals 360. Each motor will hold its position in a servo like mode once it reaches the encoder target.* |
| **Set Motor Direction Invert**<br>Inverts the forward/reverse direction mapping of a DC motor channel. This function is intended to harmonize the forward and reverse directions for motors on opposite sides of a skid-steer robot chassis. Inverting one motor channel can making coding opposite facing DC motors working in tandem more intuitive. An invert parameter of 1 = invert. An invert parameter of 0 = no invert. The default is no invert. | setMotorInvert(); | setMotorInvert(1, 1);<br>*Invert the spin direction mapping of Motor 1.*<br>setMotorInvert(2, 1);<br>*Invert the spin direction mapping of Motor 2.* |
| **Read Motor Busy Status**<br>The busy flag can be read to check on the status of a DC motor that is operating in positional PID mode. The motor busy status will return a "1" if it is moving toward a positional target (degrees or encoder count). Once it has reached its target and is in hold mode, the busy status will return a "0". | readMotorBusy(); | readMotorBusy(1);<br>*Return the busy status of Motor 1.*<br>readMotorBusy(2);<br>*Return the busy status of Motor 2.* |

| | | |
|---|---|---|
| **Read DC Motor Current**<br>Reads the DC motor current of each TETRIX DC motor attached to PRIZM Motor 1 and Motor 2 ports. The integer value that is returned is motor load current in milliamps. | readMotorCurrent(); | readMotorCurrent(1);<br>*Read the motor load current of the PRIZM Motor 1 channel.*<br>readMotorCurrent(2);<br>*Read the motor load current of the PRIZM Motor 2 channel.*<br><br>*Example: 1500 = 1.5 amps* |
| **Read Encoder Count**<br>Reads the encoder count value. The PRIZM controller uses encoder pulse data to implement PID control of a TETRIX DC motor connected to the motor ports. The PRIZM controller counts the number of pulses produced over a set time period to accurately control velocity and position. Each ¼ degree equals one pulse or "count", or 1 degree of rotation equals 4 encoder counts. The current count can be read to determine a motors shaft position. The total count accumulation can range from -2,000,000,000 to 2,000,000,000. A clockwise rotation adds to the count value while a counter-clockwise rotation subtracts from the count value. The encoder values are set to zero at power up and reset. | readEncoderCount(); | readEncoderCount(1);<br>*Read the current count value of encoder 1 (ENC 1 port).*<br>readEncoderCount(2);<br>*Read the current count value of encoder 2 (ENC 2 port).* |
| **Read Encoder Degrees**<br>Reads the encoder degrees value. The PRIZM controller uses encoder pulse data to implement PID control of a TETRIX DC motor connected to the motor ports. The PRIZM controller counts the number of pulses produced over a set time period to accurately control velocity and position. This function is similar to the encoder count function, but instead of returning the raw encoder count value, it returns the motor shaft position in degrees. The total degrees count accumulation can range from -2,000,000,000 to 2,000,000,000. A clockwise rotation adds to the count value while a counter-clockwise rotation subtracts from the count value. The encoder values are set to zero at power up and reset. | readEncoderDegrees(); | readEncoderDegrees(1);<br>*Read the current degrees count value of encoder 1 (ENC 1 port).*<br>readEncoderDegrees(2);<br>*Read the current degrees count value of encoder 2 (ENC 2 port).* |
| **Reset Encoders**<br>Calling this function will reset the encoder count accumulator to zero. | resetEncoder(); | resetEncoder(1);<br>*Reset the encoder 1 count to zero.*<br>resetEncoder(2);<br>*Reset the encoder 2 count to zero.* |

| | | |
|---|---|---|
| **Read Line Sensor Output**<br>Read the digital output of the Line Finder sensor connected to a PRIZM sensor port. The value read is "0" when reflected light is received (detecting a light colored surface), and a "1" when light is not being received (detecting a dark colored surface, i.e. line). | readLineSensor(); | readLineSensor(D2);<br>*Read the digital value of a line finder sensor on digital sensor port D2.*<br><br>*Note: The line finder can be connected to any digital port D2 – D5, or analog ports A1 – A3 configured as digital input.* |
| **Read Ultrasonic Sensor as Centimeters**<br>Read the distance in centimeters of an object placed in front of the ultrasonic sensor. The sensor is modulated at 42 kHz and has a range of 3 to 400 centimeters. The value read is an integer number. | readSonicSensor CM(); | readSonicSensorCM(D3);<br>*Read the distance in centimeters of an object placed in front of the ultrasonic sensor connected to digital sensor port D3.*<br><br>*Note: The ultrasonic sensor can be connected to any digital port D2 – D5, or analog ports A1 – A3 configured as digital input.* |
| **Read Ultrasonic Sensor as Inches**<br>Read the distance in Inches of an object placed in front of the ultrasonic sensor. The sensor is modulated at 42 kHz and has a range of 2 to 150 inches. The value read is an integer number. | readSonicSensorIN(); | readSonicSensorIN(D4);<br>*Read the distance in inches of an object placed in front of the ultrasonic sensor connected to digital sensor port D4.*<br><br>*Note: The ultrasonic sensor can be connected to any digital port D2 – D5, or analog ports A1 – A3 configured as digital input.* |
| **Read Battery Pack Voltage**<br>Read the TETRIX battery pack voltage powering the PRIZM controller. The value read is an integer number. | readBatteryVoltage(); | readBatteryVoltage();<br>*Read the TETRIX battery pack voltage powering the PRIZM controller.*<br><br>*Example: A value of 918 equals 9.18 volts.* |
| **Read Start Button State**<br>Reads the state of the green PRIZM start pushbutton. A returned value of "1" indicates a pressed state. A returned value of "0" indicates not pressed state. | readStartButton(); | readStartButton();<br>*Read the start button. A value of "1" means button is pressed. A value of "0" means button is not pressed.* |

| | | |
|---|---|---|
| **Set Speed of a Servo Motor**<br>This function sets the speed of a servo motor connected to a PRIZM servo port 1 – 6. The speed parameter can be 0 to 100 percent. The servo motor channel parameter can be any number 1 to 6. If not specified, the speed of a servo defaults to 100 (maximum speed). Once a servo speed has been set, it will always move at the set speed until changed. Unless you are changing speeds, it will only need to be called once at the beginning of a program. | setServoSpeed(); | setServoSpeed(1, 25);<br>*Set the speed of servo channel 1 to 25%.*<br>setServoSpeed(2, 50);<br>*Set the speed of servo channel 2 to 50%.* |
| **Set Speed of All Servo Motors**<br>This function will set the speed of ***all*** six servo channels simultaneously with a single command. All six speeds are in sequential order and can be 0 to 100 percent. All six servo speeds may be the same or different. | setServoSpeeds(); | setServoSpeeds(25,25,25,25,25,25);<br>*Set the speed of all six servo channels to 25%.*<br>setServoSpeeds(25,35,45,55,65,75);<br>*Servo 1 speed = 25%*<br>*Servo 2 speed = 35%*<br>*Servo 3 speed = 45%*<br>*Servo 4 speed = 55%*<br>*Servo 5 speed = 65%*<br>*Servo 6 speed = 75%* |
| **Set Position of a Servo Motor**<br>Set the angular position of a servo motor connected to a PRIZM servo motor port 1 – 6. The position parameter can be any value between 0 and 180 degrees. Any value out of this range is ignored. Not all servos are the same, so be careful when operating a servo motor at the extreme ranges. Listen closely, if a servo is buzzing, it is pressing against its mechanical stop which may damage the motor. If this happens, limit the extreme range to values slightly greater than 0 and slightly less than 180 to avoid damage to the servo motor. | setServoPosition(); | setServoPosition(1, 90);<br>*Set the angular position of servo 1 motor to 90 degrees.*<br>setServoPosition(2, 130);<br>*Set the angular position of servo 2 motor to 130 degrees.* |
| **Set Position of All Servo Motors**<br>Set the angular position of ***all*** six servo motors connected to the PRIZM servo motor ports 1 – 6. The position parameter can be any value between 0 and 180 degrees. Any value out of this range is ignored. Not all servos are the same, so be careful when operating a servo motor at the extreme ranges. Listen closely, if a servo is buzzing, it is pressing against its mechanical stop which may damage the motor. If this happens, limit the extreme range to values slightly greater than 0 and slightly less than 180 to avoid damage to the servo motor. | setServoPositions(); | setServoPositions(90,90,90,90,90,90);<br>*Set the angular position of all six servo motors connected to PRIZM servo ports 1 – 6 to 90 degrees.*<br>setServoPositions(10,20,30,40,50,60);<br>*Set the angular position of all six servo motors connected to PRIZM servo ports 1 – 6.*<br>*Servo 1 position = 10 degrees*<br>*Servo 2 position = 20 degrees*<br>*Servo 3 position = 30 degrees*<br>*Servo 4 position = 40 degrees*<br>*Servo 5 position = 50 degrees*<br>*Servo 6 position = 60 degrees* |

| | | |
|---|---|---|
| **Read a Servo Position**<br>Reads the most recent commanded position of a servo motor connected to PRIZM servo ports 1 – 6. The value returned will be 0 – 180. | readServoPosition(); | readServoPosition(1);<br>*Read the most recent commanded position of servo 1.*<br>readServoPosition(2);<br>*Read the most recent commanded position of servo 2.* |
| **Set Continuous Rotation (CR) State**<br>Set the direction of on/off and direction state of a CR servo connected to PRIZM CR1 and CR2 ports. A state parameter of -100 equals spin counter-clockwise. A state parameter of 100 equals spin in clockwise direction. A state parameter of 0 is off or "stop". Motor parameter can be a "1" or "2" commanding either CR1 or CR2 servo ports. | setCRServoState(); | setCRServoState(1, 100);<br>*Spin CR1 servo continuously clockwise.*<br>setCRServoState(1, 0);<br>*Stop CR1 servo.*<br>setCRServoState(1, -100);<br>*Spin CR1 servo continuously counter-clockwise.* |