Matrix Robotic Controller Specification

For LEGO MINDSTORMS NXT

MATRIX Robotics International Ltd

 Revision 1.3

Contents

Matrix Robotic Controller for LEGO NXT

1. Summary

The Matrix Robotic Controller is a combined DC Motor controller and servo controller for use with the Matrix Robotics Building system. It provides an interface between a LEGO NXT and the Matrix electromechanical components. The controller has 4 DC motor ports to connect motors with built in encoders and 4 servo outputs with built in voltage conversion from the 9.6v battery supply to the servo's 6v supply.

It connects to the MINDSTORMS NXT via a LEGO defined I2C interface.

Features include;

- A recessed power switch.
- A built in self resetting fuse protects against over current operation.
- A bi-color activity LED shows the overall operational status.

2. Description

2.1. Function

The Matrix controller allows a processing unit such as the LEGO MINDSTORMS NXT controller to expand a single sensor port to control four motors and four servos.

The controller also supplies the processing unit with status which includes battery voltage and current motor position for each of the four motors.

3. Technical Description

3.1. Functional LED

A bi-color activity LED is included to assist in verification of correct operation with the following status indications;

- A continuous green LED - the controller is idle.
- A blinking green LED - the controller is exchanging data with the NXT.
- A continuous red LED - the controller is driving at least one motor or servo.
- A blinking red LED - the controller is driving at least one motor or servo and that the controller is exchanging data with the NXT.

3.2. I2C interface

The controller supports an I2C interface scheme to enable it to be easily controlled by the NXT. This interface relies on an address register and an addressable memory space. The address register may be written to using an I2C write function. Subsequent bytes which follow the address will be placed in the memory starting at the location written to the address register. The address register automatically increments each time a location is written to or read from. An I2C read operation will start reading from the current address register memory address.

Address	Type	Contents
00 – 07H	chars	Controller version number
08 – 0FH	chars	Manufacturer
10 – 17H	chars	Controller type
18 – 3DH	bytes	Not used
3E, 3FH	Byte	Reserved
40H	byte	Not used
41H	byte	Controller status
42H	byte	Timeout control
43H	byte	Battery level
44H	byte	Start flag
45H	byte	Servo enable
46H	byte	Servo 1 speed
47H	byte	Servo 1 target
48H	byte	Servo 2 speed
49H	byte	Servo 2 target
4AH	byte	Servo 3 speed
4BH	byte	Servo 3 target
4CH	byte	Servo 4 speed
4DH	byte	Servo 4 target
4E – 51H	lword	Motor 1 position
52 – 55H	lword	Motor 1 target
56H	byte	Motor 1 speed
57H	byte	Motor 1 mode
58 – 5BH	lword	Motor 2 position
5C – 5FH	lword	Motor 2 target
60H	byte	Motor 2 speed
61H	byte	Motor 2 mode
62 – 65H	lword	Motor 3 position
66 – 69H	lword	Motor 3 target
6AH	byte	Motor 3 speed
6BH	byte	Motor 3 mode
6C – 6FH	lword	Motor 4 position
70 – 73H	lword	Motor 4 target
74H	byte	Motor 4 speed
75H	byte	Motor 4 mode

The *Controller version number* field will report a revision number in the format "Vn.m" where *n* is the major version number and *m* is the revision level. Revision numbers will typically reflect the firmware level. The version number will be used to indicate the hardware level.

The *Manufacturer* field will contain "HiTechnc".

The *Controller type* field will contain "M4S4cont".

The *Controller status* returns the current controller status, thus;

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	–	Batt. low	Fault

The *Timeout control* enables the automatic motor and servo shutdown to be controlled. When set to a non-zero value, the controller will automatically shut down the motors and servos if no I2C transactions are received within the specified number of seconds. If set to zero, the timeout function will be disabled. The default setting is 2.

The *Battery level* returns the current battery voltage in units of 40mV.

The *Start flag* may be used to synchronize any motor functions which are set as *pending start flag*. It should be set to 1 to initiate a change in motor function. It and all *pending start flags* will automatically reset to zero once the start has occurred.

The *Servo enable* allows the generation of servo control pulses to be controlled.

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	Servo4	Servo3	Servo2	Servo1

When a bit is set to one, the pulses for that servo are enabled. If a bit is set zero, the pulses will be disabled. The settings within this field are unaffected by the operation of the controller timeout function. The servos will be disabled when a timeout occurs, but will revert back to the *Servo enable* byte when the I2C communication resumes.

The *Servo 1/2/3/4 speed* bytes allow the rate, at which changes to the servo positions are made, to be controlled. If the value is set to zero, changes to the servo position is immediate. If the value is non-zero, changes will occur at a rate equal 10*value milliseconds per step.

The *Servo 1/2/3/4 target* bytes allow the servo pulses to be varied from 0.75mS – 2.25mS with the byte value ranging from 0 – 250.

The *Motor 1/2/3/4 position* signed long word (high byte through low byte) return the current encoder reading for the motor channel.

The *Motor 1/2/3/4 target* signed long word (high byte through low byte) allow a target position to be specified for a motor channel if it is *slew to position* mode.

The *Motor 1/2/3/4 speed* byte allows the motor channel speed to be set from -100 – 100. A value of zero (0) causes the motor to stop. The sign of the speed value controls the motor direction. In *slew to position* mode, the sign is ignored.

The *Motor 1/2/3/4 mode* byte controls the operation of the motor channel;

D7	D6	D5	D4	D3	D2	D1	D0
Busy	–	–	Invert	Pending	Reset	M1	M0

The *M0* and *M1* bytes encode the operating mode, thus;

M1 M0	Encoding
0 0	Power control only – 0 speed signifies motor float
0 1	Power control only – 0 speed signifies motor brake
1 0	Speed control
1 1	Slew to position

Setting the *Reset* bit causes the position, the target, the speed and mode fields to be cleared to 0. This will cause the motor to go into float mode. The reset bit will thus be cleared when these fields have been cleared

Setting the *Pending* bit causes changes to both the target, speed and mode fields to be delayed until the (global) *Start flag* is set to 1. The *Start flag* and all pending bits are automatically cleared to 0 after the pending changes have been initiated.

Setting the *Invert* bit causes the motor channel to invert the direction of the motor spindle which is interpreted as positive. When this bit is zero, a clockwise motor rotation, as viewed from the front of the motor, is regarded as positive. When this bit is one, a counter-clockwise motor rotation is regarded as positive. This is intended to make programming the left hand side drive motor simpler.

The *Busy* bit will be set to 1 while the Slew to position function has not yet reached the target position.

4. Operational notes

The controller has certain features which have quite specific intended uses.

4.1. Mode register *Invert* bit

The *Mode* register *Invert* bit is intended as a configuration setting, not a live motor control function. For a skid steer platform, one motor must rotate in the opposite direction to the other one for the wheels to drive the unit backward or forward. This bit is intended to enable both motors to be commanded in the same direction to obtain smooth forward and backward movement.

4.2. Mode register *Pending* bit

The *Mode* register *Pending* bit is intended as a way to synchronize a change in motor operations involving multiple motors. For a skid steer platform, both motors must start rotating at exactly the same moment to drive the unit accuratlyaccurately backward or forward. By setting this bit in each motor's *Mode* register and then setting the operational settings, the motors' new settings will be delayed from taking effect until the *Start flag* is set. This is particularly important with I2C masters as slow as the NXT.

4.3. Mode register *Reset* bit

The *Mode* register *Reset* bit is intended as a way to reset a motor channel to a known state. Note that if the motor is rotating when this function is performed, the motor will run to a halt which will likely make the *Motor position* register non zero. To over come this issue, it is highly recommended that two Reset functions be issued, the first to reset the channel and put the motor into float and the second about ½ second later to ensure that the *Motor position* is zero.

4.4. *Motor position* register

The *Motor position* register is an accumulator and not intended to be reset between motor movements. The host software, typically a program running in the NXT, should keep track of the position register if necessary.

If the motor is only being used in *Modes* 0, 1 or 2, then simple odometry can be performed by calculating the distance moved thus;

Distance moved = current position – previous position

If the motor is only being used in *Mode 3*, then the host software should maintain its own version of the position register based on the changes being made to the target register, since reading the actual position register may contain a small error as the firmware adjusts getting to the target position. If the actual position register is used, small position errors will creep in.

If the motor is transitioned from either of *Modes* 0, 1 or 2 to *Mode* 3, the host software should initialize its own version of the position register just once when the mode is changed. If this is not done and a target position is commanded which bears no relationship to the current position, the motor may slew for a considerable period of time.

4.5. *Servo target* register

The *Servo target* register can potentially attempt to move a servo beyond its internal mechanical limits. When operating at <50 or >205, listen to the servo and ensure that it is not continually trying to get beyond its internal mechanical limit. This may damage the servo, overheat the servo and run batteries down quickly.

Appendix

For more information contact Matrix Robotics at support@matrixrobotics.com

Or

HiTechnic at support@hitechnic.com