ZikoDrive ZD10UART Series Operating Manual

For Firmware V3.5 and above (dated 10/10/19)

Introduction

OverView

This manual is intended to be used with the ZikoDrive ZD SERIES UART Motor controller running standard firmware (V3.5 and above) and includes the following UART Commands that can also be used over RS232 or RS485 using the relevant Break Out Boards:

- Run at a set speed
- Run Number of Microsteps
- Decelerate to Stop
- Emergency Stop
- Move to the Home position
- Reset Home Position
- Move to Absolute Position
- Move to Home input Limit Switch
- Setting STD motor Parameters including Running Current, Holding Current, Acceleration/Deceleration Current, Acceleration/Deceleration Rates and Microstepping resolution.

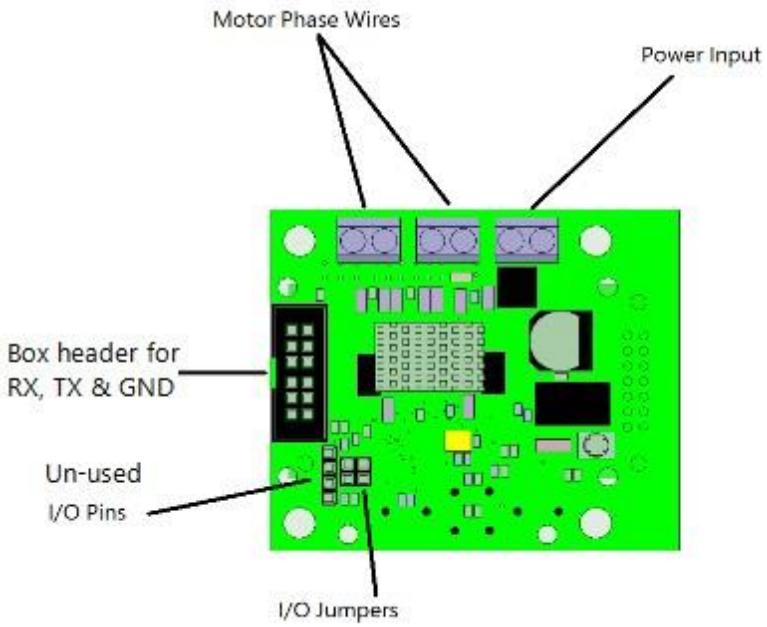
Specifications

- Up to 30Vdc.
- Up to 5A.
- Up to128 Microsteps/Step.
- Modular design connects to many Break Out Boards that can provide options such as 420mA, 0-10Vdc, Optically Isolated inputs, etc...
- Over-temperature protection and stall detection.
- Microprocessor controlled enabling exceptional positional accuracy.

Contents

Contents

Board View



Important NOTE:

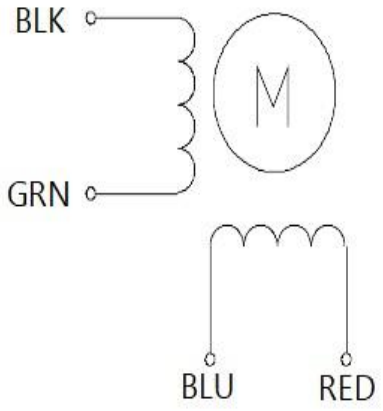
It is critically important to not exceed the recommended input voltages as this may cause unrepairable damage to the Zikodrive ZD SERIES UART Motor Driver. The max voltage recommended for any digital or analogue inputs is 5Vdc. DO NOT APPLY Higher Voltage PWM as this will damage the board.

All wires must be kept as short as possible to reduce EMC emissions from the motor phase cables and to reduce noise on both the analogue and digital inputs. If you require more information regarding good wiring practice, please contact ZikoDrive enquiries@zikodrive.com.

Wiring Motor Phase Wires

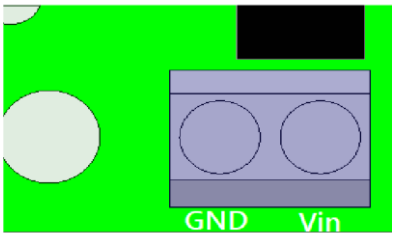
Wire up the Motor phase wiring according to the motor manufacturers datasheet. Below is a common example of an off the shelf unit. Wiring up the motor phases incorrectly will not damage the board, however it will affect the direction and performance of the motor and in some cases the motor may not run at all. As the ZD SERIES UART PCB is an absolute positioning Controller, if the motor is traveling in the incorrect direction absolute positioning commands will not work.

Terminal	Wire
A2	Black
A1	Green
B1	Blue
B2	Red



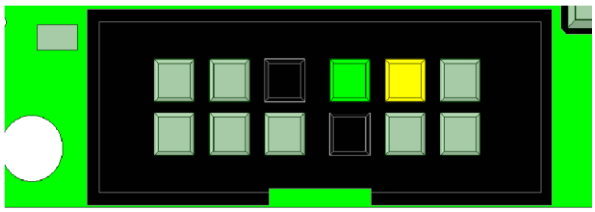
Power Wires

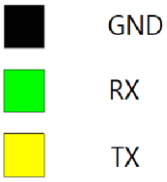
12-30Vdc must be applied to the board on the 2 Way Terminal Block Marked GND & Vin. Although the board is protected against reverse polarity, care should be taken to ensure the wires are not reversed.
Note – Please ensure that the DC power supply has the proper current supply capability. Care should be taken to ensure the DC supply is clean (keep wires as short as possible, and shield where required).



Once all these connections are complete, the ZD SERIES UART can be powered up.

Connections to Box Header (UART)





GND is the common digital GND for the PCB. The RX and the TX pins must be switched with the Master Controller. I.e., TX (transmitting) from the Master controller must be connected to the RX (receiving) pin on the ZD SERIES UART. The TX (transmitting) from the ZD SERIES UART must be connected to the RX (receiving) of the Master Controller.

Connections for any serial Break Out Board (RS485 & RS232)

Connecting to the Break Out Boards.

A = Non-inverting receiver input and non-inverting driver output
B = Inverting receiver input and inverting driver output
GND = Ground

Note - If shielded cables are used between the Master Controller and the ZikoDrive please ensure only one end of the shielding is grounded to ensure no Ground loops.

CAUTION – supplying RS485 or RS232 signals directly to the UART pins will damage the controller. If you need to use RS232 or RS485 please ensure you have the appropriate plug in board.

Programming Manual

The UART Protocol

The protocol used is non-standard and derived by the ZikoDrive engineers to provide fast and efficient data transfer.
The Data is transmitted and received in 8bit Bytes. Which, in this manual will be displayed in a Hexadecimal format.

Please note, that no bytes can be sent with a Value of 0xFF as this value can not be processed with the 8bit UART protocol.

Protocol Setup

BAUD RATE	=	19200
STOP BITS	=	One
PARITY	=	None
Bit order	=	LSB first

The ZikoDrive Protocol

All commands must be sent with a two-byte header, one-byte address code, one-byte register address code, an “x” amount of data bytes and then followed with a CheckSum.
To ensure the correct data is being transferred, the protocol will send either a confirm (0x06) or a reject (0x15) for a good, or a bad command. If the ZD SERIES UART has accepted the command, the confirmation will then be sent to the Master Controller and then process the given command. If the command was rejected, all data will be disregarded, and a new command must be sent.

Example 1 – Basic Send

From Master 0x7A 0x64 0x01 0x03 0xFB

0x7A & 0x64	=	Header, ASCII code ZD
0x01	=	ZikoDrive Address, Set as 0x01 as default
0x03	=	Register Address, STOP command
0xFB	=	CheckSum

Note – as this is a STOP command no DATA bytes are required.

CheckSum formula

Each command sent to the ZikoDrive controller must be followed with a CheckSum byte or the data will be rejected.

CHECKSUM = ((ADDRESS + REGISTER + DATA_MSB + DATA_0 + DATA_1 + DATA_LSB) XOR 0xFF) & 0xFF;

1. The Address, the required register, and all the data bytes are added together.
2. A command is run through an XOR with 0xFF
3. A command is then run through & with 0xFF

We have available an EXCEL spreadsheet to help calculate the CheckSum byte if required. We are happy to send this over on request.
Please note that the Header and the CheckSum bytes are not included in the CheckSum Formula.

On Power Up

As the controller powers up, there will be a serial command transmitted consisting of 5 bytes. These bytes ensure that the serial protocol setup is correct, and also inform the user of the Hardware and Firmware versions of the controller.

From Zikodrive		0x7A 0x64 0xA5 0x35 0x25
0x7A & 0x64	=	Header, ASCII code ZD
0xA5	=	Hardware Code
0x35	=	Firmware Code
0x25	=	CheckSum

Command Complete

When commands are used that require time to perform (i.e., RUN_STPS) a 0x00 command is sent from the ZikoDrive to the Master Controller once the action has been completed. If the command sent is a Speed command, the 0x00 will be sent once the motor has reached the desired speed. No additional commands will be accepted by the controller until the Complete command has been sent.

Example 2 – Command Complete (RUN_STPS)

From Master	0x7A	0x64	0x01	0x02	0x01	0x00	0x7D	0x00	0x7E	
From ZikoDrive										0x06
	//After RUN_STEPS has reached its final position//									
From ZikoDrive										0x00

0x7A & 0x64 = Header, ASCII code ZD

0x01 = ZikoDrive Address, Set as 0x01 as default

0x02 = Register Address, RUN_STPS

0x01 = Direction

0x00 = Data_0 (No. of steps MSB)

0x7D = Data_1

0x00 = Data_2 (No. of steps LSB)

0x7E = CheckSum

Example 3 – Command Complete (RUN_SPD)

From Master	0x7A	0x64	0x01	0x01	0x01	0x09	0xC4	0x2F
From ZikoDrive								0x06

//After RUN_SPD has completed Acceleration ONLY//

From ZikoDrive		0x00

0x7A & 0x64 =		Header, ASCII code ZD
0x01	=	ZikoDrive Address, Set as 0x01 as default
0x01	=	Register Address, DIR_SPD
0x01	=	Direction
0x09	=	Data_0
0xC4	=	Data_1
0x7F	=	CheckSum

Shortlist of Commands

Register	Name	Notes	DATA Size (Bits)	Defaults
0x01[1]	RUN_SPD	Run at Speed	17	NA
0x02[2]	RUN_STPS	Run No. of Steps	23	NA
0x03[2]	STOP	Decelerate to STOP	0	NA
0x04[2]	EMER_STOP	Immediate STOP	0	NA
0x05[2]	MOV_HOME	Go to home position	0	NA
0x06	RST_HOME	Reset Home position	0	NA
0x07[2]	MOVE_ABS	Move to Absolute position	22	NA
0x08[2]	MOV_HOME_RST	Move Home then re-set on switch	17	NA
0x09	RUN_CUR	Running Current Setting	8	0x52
0x0A	HOLD_CUR	Holding Current Setting	8	0x00
0x0B	ACC_CUR	Acceleration Current Setting	8	0x52
0x0C	DEC_CUR	Deceleration Current Setting	8	0x52
0x0D	ACC_VAL	Acceleration Rate	12	0x085
0x0E	DEC_VAL	Deceleration Rate	12	0x085
0x0F	SPD_RUN	Set running Speed	16	0x09BA
0x10	M_STEP	Set MicroStep resolution	4	0x07
0x11	READ_PAR	Read Parameter	8	NA

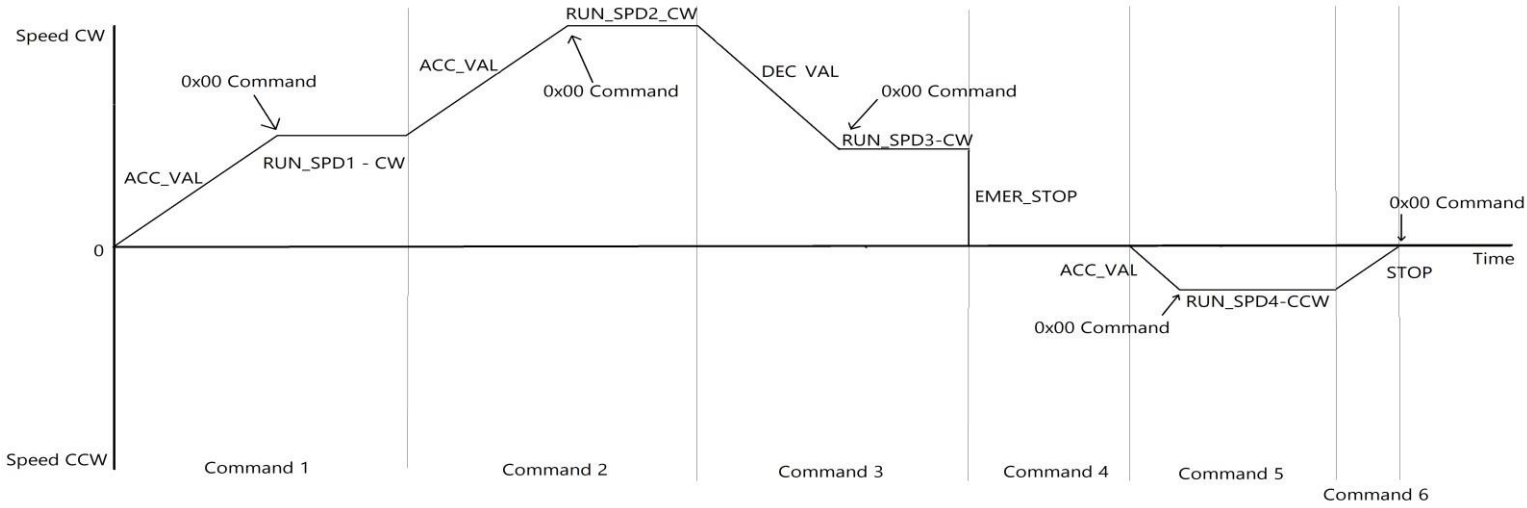
[1] 0x00 sent after initial speed transition (either acceleration or deceleration period)

[2] 0x00 sent after the whole command is completed

Incremental Commands

These commands will work from any position and does not reference itself to a fixed home position. They will run for either an infinite time in one or the other direction or several steps in either direction.

RUN_SPD (0x01)



Graph 1 - RUN_SPD, STOP & EMER_STOP

Between each speed transition (RUN_SPD1, RUN_SPD2...) the ZikoDrive controller will run the ACC_VAL and DEC_VAL values and using the current variables set in registers ACC_CUR and DEC_CUR. The EMER_STOP has no deceleration where the STOP command will decelerate the motor smoothly to a STOP using the DEC_VAL.

The RUN_SPD command sets the motor to run at a requested speed. The RUN_SPD requires 17 bits of data broken down into 3 Bytes.

- The first Data Byte is the is read as a boolean 1 or 0 that dictates the motor run direction.
- Byte_MSB is an 8bit Byte that uses the MSB of the required speed value
- Byte_LSB is an 8bit Byte that uses the LSB of the required speed value

The RUN_SPD command will accelerate the motor at the acceleration rate stated in the ACC_VAL until the required speed is achieved. Once the required speed is achieved a 0x00 completion command will be sent from the ZikoDrive.

RUN_SPD Formula

The values for the MSB and LSB required for the RUN_SPD are just simply the required motor Speed in RPM x 10 to obtain a Speed resolution of 0.1RPM.

Example 4 - RUN_SPD (Multiple Commands)

So, for a required Speed of 120RPM;

120 x 10 = 1,200
1,200 Decimal = 0x04B0 in Hexadecimal

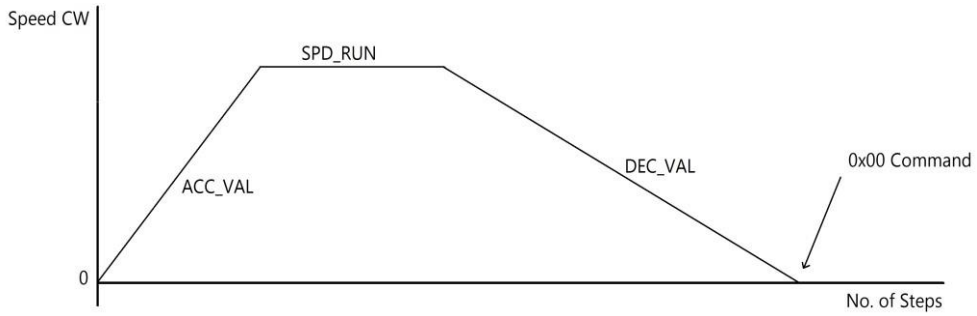
RUN_SPD_MSB = 0x04
RUNSPD_LSB = 0xB0

From Master	0x7A	0x64	0x01	0x01	0x01	0x04	0xB0	0x48	
From ZikoDrive									0x06
From Master	0x7A	0x64	0x01	0x01	0x01	0x03	0xE8	0x11	

The highlighted command will not be accepted as 0x00 completion command was not sent. I.e., the motor was still accelerating when this command was sent.

0x7A & 0x64	=	Header, ASCII code ZD
0x01	=	ZikoDrive Address, Set as 0x01 as default
0x01	=	Register Address, RUN_SPD
0x01	=	Direction
0x04	=	Speed MSB
0xB0	=	Speed LSB
0x48	=	CheckSum

RUN_STPS (0x02)



Graph 2 - RUN_STPS CW

The RUN_STPS command sets the motor to run a set number of Microsteps in the required direction. The RUN_STPS requires 23 bits of data broken down into 4 Bytes.

- The first Data Byte is read as a boolean 1 or 0 that dictates the motor run direction.
- Byte_MSB is an 8bit Byte that uses the MSB of the required Microstep value
- Byte_0 is an 8bit Byte that uses the LSB of the required Microstep value
- Byte_LSB is an 8bit Byte that uses the LSB of the required Microstep value

The RUN_STPS command will accelerate the motor at the acceleration rate stated in the ACC_VAL until the SPD_RUN speed is achieved. The motor will then maintain this speed until deceleration at the DEC_VAL rate and then stop when the number of Microsteps is achieved, and then a 0x00 completion command will be sent from the ZikoDrive.
Any command sent to the ZikoDrive before the 0x00 completion command is sent will be disregarded.

The Microstep size is determined by the value stored in the M_STEP register (please read the M_STEP register for more details).

RUN_STPS Formula

The values for the Data Bytes required for the RUN_STPS are the required number of Microsteps.

Example 5 - Setting RUN_STPS

So, M_STEP was set to 128 Microsteps/Step there would be 25,600 steps per revolution. So, if we needed to rotate 16.2 revolutions we need:

25600 x 16.2 = 414,720 Microsteps
414,720 Decimal = 0x065400 Hexadecimal

RUN_STPS_MSB = 0x06
RUN_STPS_MID = 0x54
RUNSTPS_LSB = 0x00

From Master	0x7A	0x64	0x01	0x02	0x01	0x06	0x54	0x00	0xA1	
From ZikoDrive										0x06

0x7A & 0x64 = Header, ASCII code ZD

0x01 = ZikoDrive Address, Set as 0x01 as default

0x02 = Register Address, RUN_STPS

0x01 = Direction

0x06 = Speed MSB

0x54 = Speed MID

0x00 = Speed LSB

0xA1 = CheckSum

General Commands

These commands will work when running both Incremental commands or absolute commands.

STOP (0x03)

The STOP command, typically used after the RUN_SPD command will bring the motor to a dead stop using the DEC_VAL. Once stopped the ZikoDrive will apply the HOLD_CUR current to ensure the motor does not move from externally applied loads.

Running the STOP Command when the HOLD_CUR value = 0x00 will put the driver into a high impedance state.

Example 6 - STOP

From Master	0x7A	0x64	0x01	0x03	0xFA		
From ZikoDrive						0x06	0x00

EMER_STOP (0x04)

The EMER_STOP command will stop the motor immediately with no deceleration and then apply the HOLD_CUR to hold the motor stationary.

Running the EMER_STOP Command when the HOLD_CUR value = 0x00 will put the driver into a high impedance state.

Example 7 - EMER_STOP

From Master	0x7A	0x64	0x01	0x04	0xF9		
From ZikoDrive						0x06	0x00

Absolute Commands

The absolute commands are completed with a reference to a fixed home position. Ie, moving to an absolute number of steps from the home position.
Care must be taken when using the Absolute commands that the M_STEP register is not changed during the use of the Absolute Commands. This is because the ZikoDrive uses the M_STEP resolution as an incremental counter from the HOME position and when the M_STEP register changes the counter becomes meaningless.

MOV_HOME (0x05)

The MOV_HOME command will take the motor in either CW or CCW (it will take the shortest path) to the set home position. The home position is the position the motor is in when the board powers up, or the position set using either the RST_HOME command or the MOV_HOME_RST command. The MOV_HOME command will use the ACC_VAL to accelerate up to the SPD_RUN speed then decelerate using the DEC_VAL down and then stop on the HOME position.

Example 8 - MOV_HOME & High Impedance

From Master	0x7A	0x64	0x01	0x05	0xF8					
From ZikoDrive						0x06				
//After MOV_HOME has completed stopped on the HOME position//										
From ZikoDrive						0x00				
From Master							0x7A	0x64	0x01	0x03	0xFA
From ZikoDrive											0x00

RST_HOME (0x06)

The RST_HOME command will re-set the home position to ZERO, making the current position of the motor the new HOME position.

Example 9 - RST_HOME

From Master	0x7A	0x64	0x01	0x06	0xF7		
From ZikoDrive						0x06	0x00

MOV_ABS (0x07)

The MOV_ABS command will rotate the motor in the requested direction to a specified number of steps from the HOME position. The MOV_ABS also uses the ACC_VAL to accelerate, SPD_RUN value as constant speed and DEC_VAL to decelerate and stop on the required position. The MOV_ABS is always in agreement with the selected M_STEP register Microsteps.

The MOV_ABS position register contains 21bits of positional data in each direction from the home position. The MSB (of the entire 22bits) is used as a negative sign – IE, two's complement format.

So, this means that if a step resolution of 16 Microsteps per step was used, the absolute Max positive and negative positions would be:

128 Microsteps = 3,200 steps/rev
21bits = decimal 2,097,151
2,097,151 / 3,200 = 655.36 revolutions in each direction (with 1.8 degree stepper motor)

Example 10 - MOV_ABS

If we suggest we are in a 1/16 Microstep Mode, and currently at any position CW from the requested position (let's say 6400 steps). Meaning the motor will automatically move CCW to our required position.

6400 decimal = 0x001900 in HEX giving:

MSB = 0x00
MID = 0x19
LSB = 0x00

From Master	0x7A	0x64	0x01	0x07	0x00	00x19	0x00	0xDE	
From ZikoDrive								0x06	
From ZikoDrive									0x00

0x7A & 0x64 = Header, ASCII code ZD

0x01 = ZikoDrive Address, Set as 0x01 as default

0x07 = Register Address, MOV_ABS

0x00 = MOV_ABS MSB

0x19 = MOV_ABS MID

0x00 = MOV_ABS LSB

0xDE = CheckSum

0x00 = Command Complete

MOV_HOME_RST (0x08)

The MOV_HOME_RST command will run the motor in the required direction and accelerate up to the required speed set in the MSB and LSB bytes using the ACC_VAL and continue at this speed until a limit switch is triggered. Once the limit switch has been triggered the driver will then re-set the home position to ZERO making the current position the new HOME position.

Example 11 - MOV_HOME_RST

So, for a required Speed of 10RPM CCW until Switch is triggered;

10 x 10 = 100
100 Decimal = 0x64 in Hexadecimal

RUN_SPD_MSB = 0x00
RUNSPD_LSB = 0x64

From Master 0x7A 0x64 0x01 0x08 0x00 0x64 0x00 0x92
From ZikoDrive 0x06
//After MOV_HOME_RST has completed stopped and new position SET//
From ZikoDrive 0x00

Parameter Setup

The parameter setup includes motor criteria that the ZikoDrive controller needs to know for the motor to function correctly within the application.

IMPORTANT NOTE: Care must be taken when setting these parameters as if any of these parameters were to be set up incorrectly unrepairable damage can be caused to either the motor, the ZikoDrive controller or other equipment connected to the motor.

Motor Current

There are four different modes in which the motor current can be set (during acceleration, normal running, deceleration, and when stationary). If the Current in each mode is set too high, there is high risk of unrepairable damage to all components involved.

The current set by the motor corresponds to the amount of torque the motor will have. However, may become less relevant at higher motor speeds (due to motor inductance). The recommended Phase current will be stated in all Stepper motor datasheets. Normally attempting to increase the current beyond the stated phase current will only saturate the motor's magnetic field and no additional torque can be given. However, keeping the current as low as possible will improve motor efficiency and keep the temperature of the motor and ZikoDrive down.

PLEASE NOTE – THE CURRENT VALUES MUST BE CHANGED IF CHANGING BETWEEN STANDARD AND PRECISION MICROSTEPPING. THE PHASE CURRENTS USE DIFFERENT FORMULAE, GETTING THE VALUES INCORRECT MAY CAUSE UNREPAIRABLE DAMAGE TO THE MOTOR OR ZIKODRIVE.

Standard Mode Current formula:

$$\text{Current (A)} = (\text{DATA} \times 0.26) + 0.26$$

Where DATA is the value inserted into one of the current registers.

Example 12 - Standard Mode Current Calculation

If the required current is 3.8A:

3.8 – 0.26 = 3.54
3.54 / 0.26 = 13.6

So, a value of 13 (always round down) to be entered into the required current register.

From Master	0x7A	0x64	0x01	0x09	0x0D	0xE8		
From ZikoDrive							0x06	0x00

Precision Mode Current formula:

$$\text{Current (A)} = (Vs \times DATA)/(Rm \times 256)$$

Where DATA is the value inserted into one of the current registers.
Vs is the supply voltage applied to the ZikoDrive controller.
Rm is the Motor Phase resistance (can be found on the motor manufacturers datasheet)

Example 13 - Precision Mode Current Calculation

If the required current is 3.8A, the motor resistance is 1.65 Ohms and the supply voltage is 24V:

Rm x 256 = 422.4
3.8 x 422.4 = 1,605.12
1,605.12 / 24 = 66.88

So, a value of 66 (always round down) to be entered into the required current register.

From Master	0x7A	0x64	0x01	0x09	0x42	0xB3		
From ZikoDrive							0x06	0x00

RUN_CUR (0x09)

RUN_CUR is the current used during normal running conditions (no acceleration or deceleration). The value inserted into this register should be equal to or smaller than the phase current stated on the stepper motor manufacturers datasheet.

HOLD_CUR (0x0A)

HOLD_CUR is the current used whenever the motor becomes stationary. A low value allows the motor to have a lower holding torque allowing the motor shaft to be moved under the application loading or external forces. A higher value will hold the motor shaft more firmly in place. Typical values are “0” when incremental commands are used, and up to around 30% of the phase current when absolute positioning is used.

PLEASE NOTE - When running Standard Mode, even a value of 0x00 in the HOLD_CUR will not completely turn the motor current off. To ensure the ZikoDrive outputs high impedance state you must run a STOP or an EMER_STOP command.

ACC_CUR (0x0B)

ACC_CUR is the current used during motor acceleration. Typically, many applications have a larger static torque than running torque (i.e., the amount of torque required to start is higher than the amount of torque required to run the application). If this value is too low the motor will stall and no movement will occur.

DEC_CUR (0x0C)

DEC_CUR is the current used during motor deceleration. Typically, this value is the same as the Running Current, however, if precise positioning is required this value may be increased to ensure no Microsteps are lost making the positioning of the motor unreliable and inaccurate.

ACC_VAL (0x0D)

The ACC_VAL is the rate of acceleration used when any movement of the motor is performed. Acceleration is important for stepper motors as a movement that is too abrupt will stall the motor. For this reason, Zikodrive has ensured that for any speed change made a smooth transition is performed. Every time the ACC_VAL is implemented it assumes the ACC_CUR acceleration current.

ACC_VAL Formula

RPM/sec^2 = ACC_VAL x 13.744

DEC_VAL (0x0E)

The DEC_VAL is the rate of deceleration used when any movement of the motor is performed. Deceleration is important for stepper motor positioning as if the deceleration was too quick, the inertia of the rotor (and motor load) may not stop immediately and may overshoot. Every time the DEC_VAL is implemented it assumes the DEC_CUR deceleration current.

DEC_VAL Formula

RPM/sec^2 = DEC_VAL x 13.744

Example 14 - DEC_VAL

If the required deceleration is roughly $1900RPM/s^2$:

1900/13.744 = 138.2

So, a value of 138 to be entered into the DEC_VAL register.

138 Decimal = 0x8A Hexadecimal

From Master	0x7A	0x64	0x01	0x0E	0x8A	0x66		
From ZikoDrive							0x06	0x00

SPD_RUN (0x0F)

The SPD_RUN register holds the Nominal Speed value used when one of the following commands are requested:

- RUN_STPS
- MOV_HOME
- MOV_ABS

The SPD_RUN pre-set speeds are not as accurate as the other speed commands used in the RUN_SPD or MOV_HOME_RST commands and the speed should not be expected to be 100% accurate.

Example 15 - SPD_RUN

If we needed the nominal running speed of the commands stated above to be 320RPM:

320 x 10 = 3,200

3,200 Decimal = 0x0C80 Hexadecimal

SPD_RUN MSB = 0x0C
SPD_RUN LSB = 0x80

From Master	0x7A	0x64	0x01	0x0F	0x0C	0x80	0x63		
From ZikoDrive								0x06	0x00

0x7A & 0x64 = Header, ASCII code ZD

0x01 = ZikoDrive Address, Set as 0x01 as default

0x0F = Register Address, SPD_RUN

0x0C = SPD_RUN MSB

0x80 = SPD_RUN LSB

0x63 = CheckSum

0x00 = Command Complete

M_STEP (0x10)

The M_STEP register carries the value of the Microstepping resolution carried out by the ZikoDrive controller. The resolutions can be set between full step and 128 Microsteps per step.

The following commands depending on the Microstep resolution for correct step counting and absolute positioning:

- RUN_STPS
- MOV_HOME
- MOV_ABS

Please Note – Changing the M_STEP at any time will put the motor outputs into a high impedance state (same as setting HOLD_CUR = 0x00).

IMPORTANT – DO NOT CHANGE THE MICROSTEP RESOLUTION DURING ABSOLUTE POSITIONING SEQUENCE AS DOING SO WILL MAKE THE DRIVER LOSE POSITION.

VERY IMPORTANT IF THE MODE IS CHANGED FROM PRECISION MODE TO STANDARD MODE, *YOU MUST* MODIFY ALL THE CURRENT SETTINGS OR EXCESSIVE CURRENT MAY PERMANENTLY DAMAGE THE ZIKODRIVE OR THE MOTOR...

Precision Mode

The precision mode allows the ZikoDrive controller to drive stepper motors with a very smooth and very accurate position ability with 128 Microsteps per step resolution. The precision mode operates by sending a PWM voltage into each motor phase to obtain a high resolution. However, as this mode does not directly monitor the motor current, the motor may be more susceptible to motor resonance.

This Mode is more suited for lower speed (a couple of hundred RPM) high precision applications.

Standard Mode

The standard running mode of the motor is more reliable than the high precision mode, however it only has a max Microstep resolution of 16 Microsteps per step. This Mode uses an advanced current detection algorithm that implements an average current per phase rather than the more common Max current per phase. The Standard Mode operation is more resilient to stepper motor resonance and can, therefore, run motors through the dreaded “resonance frequency” range with a reduced potential for motor stall.

This mode is more suited for higher speed applications that require less positional accuracy.

M_STEP register layout

The M_STEP register contains only 4bits to transition between Precision mode and Standard Mode, as well as to specify the Microstep resolution.

M_STEP value	Mode	Microstep Resolution
0x00	Precision Mode	Full Step
0x01		Half Step
0x02		1/4 Step
0x03		1/8 Step
0x04		1/16 Step
0x05		1/32 Step
0x06		1/64 Step
0x07		1/128 Step
0x08	Standard Mode	Full Step
0x09		Half Step
0x0A		1/4 Step
0x0B		1/8 Step
0x0C		1/16 Step

Example 16 - M_STEP

If we are looking to run Precision mode 128 Microsteps/step:

From table value = 0x07

From Master	0x7A	0x64	0x01	0x10	0x07	0xE7		
From ZikoDrive							0x06	0x00

NOTE - BE SURE TO CHANGE THE CURRENT AS WELL WHEN CHANGING BETWEEN PRECISION AND STANDARD MODES!!!

READ_PAR (0x11)

The READ_PAR register is used to read the real-time values in any of the other registers. They can be used when the motor is running or stationary.

Each Parameter is set as a sub-register with the following layout.

READ_PAR register layout

Register Address	Parameter Address	Name	Return No. of Bytes	Notes
READ_PAR 0x11	0x01	READ_SPD	2x8	Read Motor Speed
	0x02	READ_POS	3x8	Read Absolute Position
	0x03	READ_RUN_CUR	1x8	Read Running Current
	0x04	READ_HOL_CUR	1x8	Read Holding Current
	0x05	READ_ACC_CUR	1x8	Read Acceleration Current
	0x06	READ_DEC_CUR	1x8	Read Deceleration Current
	0x07	READ_ACC_VAL	1x8	Read Acceleration Value
	0x08	READ_DEC_VAL	1x8	Read Deceleration Value
	0x09	READ_SPD_RUN	2x8	Read Running Speed
	0x0A	READ_M_STEP	1x8	Read Microstep resolution & Mode

READ_SPD (0x11 0x01)

The READ_SPD register can be used to read the instant running speed of the motor at any time.

Example 17 - READ_SPD

From Master	0x7A	0x64	0x01	0x11	0x01	0xEC
From ZikoDrive	0x7A	0x64	0xBB	0x01	0x09	0xBA 0x80

0x7A & 0x64 = Header, ASCII code ZD
0xBB = Device Address, Set as 0xBB as default
0x01 = Register Address, READ_PAR
0x09 = READ_SPD MSB
0xBA = Read_SPD LSB
0x80 = CheckSum

Please note that on the READ_PAR register there is no 0x00 command complete nor a 0x06 communication confirmation.

0x09BA in Hexadecimal gives 2490 in Decimal. Just like in writing the speed we multiply by 10, when reading the speed, we need to divide by 10.

So, our speed in RPM is 2490 / 10 = 249RPM

READ_POS (0x11 0x02)

The READ_POS register can be used to read the instant absolute position of the motor at any time.

Example 18 - READ_POS

From Master	0x7A	0x64	0x01	0x11	0x02	0xEA		
From ZikoDrive	0x7A	0x64	0xBB	0x02	0x0A	0x0B	0x0C	0x21

0x7A & 0x64 = Header, ASCII code ZD

0xBB = Device Address, Set as 0xBB as default

0x02 = Register Address, READ_PAR

0x0A = READ_SPD MSB

0x0B = Read_SPD

0x0C = Read_SPD LSB

0x21 = CheckSum

0x0A0B0C Decimal = 658188 in decimal

With 1/16 Microstep resolution we have 3200 steps/rev so the absolute position is 658188 / 3200 = 205.684 revolutions in the CW direction.
The READ_POS works the same as the MOV_ABS register in its format, ie The MSB (of the entire 22bits) is used as a negative sign – IE, two's complement format.

READ_***_CUR

The resisters listed below can all be read when using the correct parameter address.

- READ_RUN_CUR
- READ_HOL_CUR
- READ_ACC_CUR
- READ_DEC_CUR

When reading the current in each register, we must first know the M_STEP value to use the correct current transformation formula. IE, the Standard and the Precision mode both use different formula to translate the values in the current registers into an understandable AMPERS reading.

READ_***_CUR formula for Standard Mode

Current (A) = (DATA x 0.26) + 0.26

Where DATA is the value inserted into one of the current registers.

Example 19 - READ_RUN_CUR (Standard Mode)

From Master	0x7A	0x64	0x01	0x11	0x03	0xEA
From ZikoDrive	0x7A	0x64	0xBB	0x03	0x0C	0x35

0x7A & 0x64 = Header, ASCII code ZD
0xBB = Device Address, Set as 0xBB as default
0x03 = Register Address, READ_PAR
0x0C = Data
0x35 = CheckSum

0x0C in Hexadecimal gives 12 in Decimal.

12 x 0.26 = 3.12
3.12 + 0.26 = 3.38A

READ_***_CUR formula for Precision Mode:

Current (A) = (Vs x DATA)/(Rm x 256)

Where DATA is the value inserted into one of the current registers.
Vs is the supply voltage applied to the ZikoDrive controller.
Rm is the Motor Phase resistance (can be found on the motor manufacturers datasheet)

Example 20 - READ_ACC_CUR (Precision Mode)

From Master	0x7A	0x64	0x01	0x11	0x05	0xE8
From ZikoDrive	0x7A	0x64	0xBB	0x05	0x29	0x16

0x7A & 0x64	=	Header, ASCII code ZD
0xBB	=	Device Address, Set as 0xBB as default
0x05	=	Register Address, READ_PAR
0x29	=	Data
0x16	=	CheckSum

Assuming we are running on 24V with a motor phase resistance of 1.1 Ohms

0x29 in Hexadecimal gives 41 in Decimal.

Vs x 41 = 24 x 41 = 984
Rm x 256 = 1.1 x 256 = 281.1

984 / 281.1 = 3.5A

READ_ACC/DEC_VAL (0x11 0x07/8)

This register refers to reading the values of the acceleration and the deceleration registers.

The formula is as follows:

RPM/sec^2 = ACC_VAL x 13.744

Example 21 - READ_ACC_VAL

From Master	0x7A	0x64	0x01	0x11	0x07	0xE6	
From ZikoDrive	0x7A	0x64	0xBB	0x07	0x00	0x8A	0xB3

0x7A & 0x64 = Header, ASCII code ZD

0xBB = Device Address, Set as 0xBB as default

0x07 = Register Address, READ_PAR

0x00 = Data MSB

0x8A = Data LSB

0xB3 = CheckSum

0x008A Hexadecimal gives 138 in binary

RPM/sec^2 = 138 x 13.744 = 1896 RPM/sec^2

READ_SPD_RUN (0x11 0x09)

The SPD_RUN register holds the Nominal Speed value used when one of the following commands are requested:

- RUN_STPS
- MOV_HOME
- MOV_ABS

Example 22 - READ_SPD_RUN

From Master	0x7A	0x64	0x01	0x11	0x09	0xE4	
From ZikoDrive	0x7A	0x64	0xBB	0x09	0x09	0x60	0xD2

0x7A & 0x64 = Header, ASCII code ZD

0xBB = Device Address, Set as 0xBB as default

0x09 = Register Address, READ_PAR

0x09 = Data MSB

0x60 = Data LSB

0xD2 = CheckSum

0x0960 in Hexadecimal gives 2400 decimal

2400 / 10 gives 240RPM

READ_M_STEP (0x11 0x0A)

The M_STEP register contains only 4bits to transition between Precision mode and Standard Mode, as well as to specify the Microstep resolution.

M_STEP value	Mode	Microstep Resolution
0x00	Precision Mode	Full Step
0x01		Half Step
0x02		1/4 Step
0x03		1/8 Step
0x04		1/16 Step
0x05		1/32 Step
0x06		1/64 Step
0x07		1/128 Step
0x08	Standard Mode	Full Step
0x09		Half Step
0x0A		1/4 Step
0x0B		1/8 Step
0x0C		1/16 Step

Example 23 - READ_M_STEP

From Master	0x7A	0x64	0x01	0x11	0x0A	0xE3
From ZikoDrive	0x7A	0x64	0xBB	0x0A	0x60	0xD2

From the table, you can see that 0x06 corresponds to Precision mode on a 1/64 resolution Microstep resolution.

NOTE - BE SURE TO CHECK THE CURRENT AS WELL THE CURRENT VALUES REQUIRE DIFFERENT FORMULA...

Quick Commands

RUN_SPD

Run 10RPM CW	0x7A 0x64 0x01 0x01 0x01 0x00 0x64 0x98
Run 10RPM CCW	0x7A 0x64 0x01 0x01 0x00 0x00 0x64 0x99
Run 100RPM CW	0x7A 0x64 0x01 0x01 0x01 0x03 0xE8 0x11
Run 100RPM CCW	0x7A 0x64 0x01 0x01 0x00 0x03 0xE8 0x12
Run 250RPM CW	0x7A 0x64 0x01 0x01 0x01 0x09 0xC4 0x2F
Run 250RPM CCW	0x7A 0x64 0x01 0x01 0x00 0x09 0xC4 0x30
Run 350RPM CW	0x7A 0x64 0x01 0x01 0x01 0x0D 0xAC 0x43
Run 350RPM CCW	0x7A 0x64 0x01 0x01 0x00 0x0D 0xAC 0x44
Run 500RPM CW	0x7A 0x64 0x01 0x01 0x01 0x13 0x88 0x61
Run 500RPM CCW	0x7A 0x64 0x01 0x01 0x00 0x13 0x88 0x62

RUN_STPS

3200 CW	0x7A 0x64 0x01 0x02 0x01 0x00 0x0C 0x80 0x6F
3200 CCW	0x7A 0x64 0x01 0x02 0x00 0x00 0x0C 0x80 0x70
32000 CW	0x7A 0x64 0x01 0x02 0x01 0x00 0x7D 0x00 0x7E
32000 CCW	0x7A 0x64 0x01 0x02 0x00 0x00 0x7D 0x00 0x7F
320000 CW	0x7A 0x64 0x01 0x02 0x01 0x04 0xE2 0x00 0x15
320000 CCW	0x7A 0x64 0x01 0x02 0x00 0x04 0xE2 0x00 0x16

RUN_ABS

9600 CW	0x7A 0x64 0x01 0x07 0x01 0x00 0x25 0x80 0x51
9600 CCW	0x7A 0x64 0x01 0x07 0x00 0x00 0x25 0x80 0x52

STOP

STOP	0x7A 0x64 0x01 0x03 0xFB
EMER_STOP	0x7A 0x64 0x01 0x04 0xFA

HOME

MOV_HOME	0x7A 0x64 0x01 0x05 0xF9
RST_HOME	0x7A 0x64 0x01 0x06 0xF8
MOV_HOME_RST	0x7A 0x64 0x01 0x08 0xF6

Current

RUN_CUR

RUN_CUR = 8	0x7A 0x64 0x01 0x09 0x08 0xED
RUN_CUR = 12	0x7A 0x64 0x01 0x09 0x0C 0xE9
RUN_CUR = 41	0x7A 0x64 0x01 0x09 0x29 0xCC
RUN_CUR = 82	0x7A 0x64 0x01 0x09 0x52 0xA3

HOL_CUR

HOL_CUR = 0	0x7A 0x64 0x01 0x0A 0x00 0xF4
HOL_CUR = 1	0x7A 0x64 0x01 0x0A 0x01 0xF3
HOL_CUR = 12	0x7A 0x64 0x01 0x0A 0x0C 0xE8

ACC_CUR

ACC_CUR = 1	0x7A 0x64 0x01 0x0B 0x01 0xF2
ACC_CUR = 12	0x7A 0x64 0x01 0x0B 0x0C 0xE7
ACC_CUR = 41	0x7A 0x64 0x01 0x0B 0x29 0xCA
ACC_CUR = 82	0x7A 0x64 0x01 0x0B 0x52 0xA1

DEC_CUR

DEC_CUR = 1	0x7A 0x64 0x01 0x0C 0x01 0xF1
DEC_CUR = 12	0x7A 0x64 0x01 0x0C 0x0C 0xE6
DEC_CUR = 41	0x7A 0x64 0x01 0x0C 0x29 0xC9
DEC_CUR = 82	0x7A 0x64 0x01 0x0C 0x52 0xA0

ACC & DEC

ACC_VAL

ACC = 5	0x7A 0x64 0x01 0x0D 0x00 0x05 0xEC
ACC = 138	0x7A 0x64 0x01 0x0D 0x00 0x8A 0x67
ACC = 3000	0x7A 0x64 0x01 0x0D 0x0B 0xB8 0x2E

DEC_VAL

DEC = 5	0x7A 0x64 0x01 0x0E 0x00 0x05 0xEB
DEC = 138	0x7A 0x64 0x01 0x0E 0x00 0x8A 0x66
DEC = 3000	0x7A 0x64 0x01 0x0E 0x0B 0xB8 0x2D

RUN_SPD

30RPM	0x7A 0x64 0x01 0x0F 0x01 0x2C 0xC2
60RPM	0x7A 0x64 0x01 0x0F 0x05 0x58 0x92
120RPM	0x7A 0x64 0x01 0x0F 0x04 0xB0 0x3B
240RPM	0x7A 0x64 0x01 0x0F 0x09 0x60 0x86
360RPM	0x7A 0x64 0x01 0x0F 0x0E 0x10 0xD1

M_STEP

1/16 Standard Mode	0x7A 0x64 0x01 0x10 0x0C 0xE2
1/127 Precision Mode	0x7A 0x64 0x01 0x10 0x07 0xE7

READ_PAR

READ_SPD	0x7A 0x64 0x01 0x11 0x01 0xEC
READ_POS	0x7A 0x64 0x01 0x11 0x02 0xEB
READ_RUN_CUR	0x7A 0x64 0x01 0x11 0x03 0xEA
READ_HOL_CUR	0x7A 0x64 0x01 0x11 0x04 0xE9
READ_ACC_CUR	0x7A 0x64 0x01 0x11 0x05 0xE8
READ_DEC_CUR	0x7A 0x64 0x01 0x11 0x06 0xE7
ACC_VAR	0x7A 0x64 0x01 0x11 0x07 0xE6
DEC_VAR	0x7A 0x64 0x01 0x11 0x08 0xE5
READ_SPD_RUN	0x7A 0x64 0x01 0x11 0x09 0xE4
READ_M_STEP	0x7A 0x64 0x01 0x11 0x0A 0xE3

Document versions

15/10/18 New document formatted for V3.0 UART Firmware.
10/10/19 Added additional details