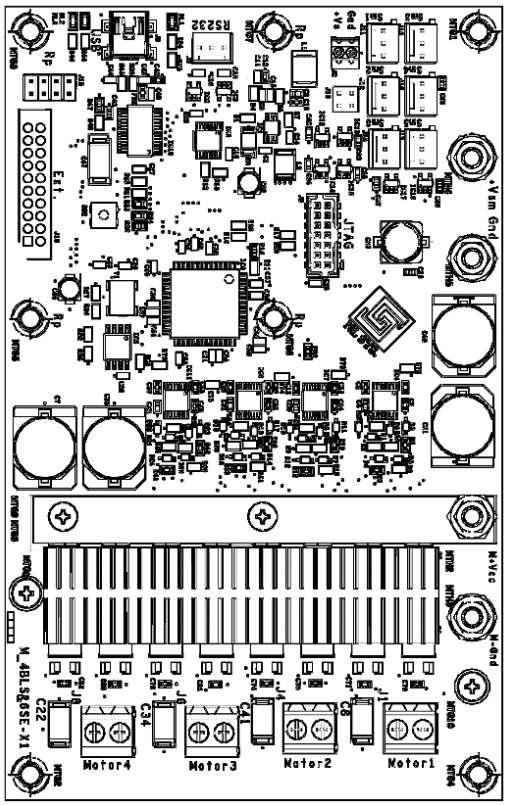
Natrolite Systems

Instruction Manual



Firmware Release **NL-A101**

HW Release **NL-022-0406-01**

M-4B6SM-X1

6 Servo Motors +
4 Brushed DC Motors Driver-Controller

Rev. 2022-A1

Table of Contents:

Preface:

This document may not be copied, or other ways reproduced, except as specifically permitted by US copyright law, without the prior written consent of Natrolite Systems. All registered trademarks are property of their respective holders.

Product options and specifications subject to change without notice. The information in this manual is furnished for information only, is subject to change without notice, and should not be construed as a commitment by Natrolite Systems. Natrolite Systems assumes no responsibility or liability for any errors or inaccuracies that mat appear in this publication.

Warning!

This board does not have any overcurrent or overvoltage protection. User must insure that attached Power supply, DC motors and Servo motors do not exceed given board specifications.

Warning!

Certain parts of the board may get hot enough to burn or cause injury during use. The user should let the board cool down before handling it. The board does not have thermal protection and may need addition cooling if it is used in extreme temperature conditions or tightly enclosed spaced.

Warning!

This board is FPGA based, with no encryption, it is possible to modify the FPGA design with new code. Doing so may cause permanent damage to the board, as well as other connected components.

About the Manual:

This manual describes the features, specifications, and programing options of M-4B6SM-X1.

The physical appearance of the M-4B6SM-X1 shown in this manual represents the use of a particular platform and version of the M-4B6SM-X1. They are provided for reference only. Actual appearance may differ depending on the type of platform used and the version of the 4B6SM-X1 installed.

Introduction

The M-4B6SM-X1 is a programable servo motor and brushed DC motor controller-driver. Each board can control up to 6 servo motors and drive up to 4 brushed DC motors. All features and controls are accessed over the USB2RS232 bridge or using a 1.2V to 5V RS232 interface. Servo motors have individually set positions, or velocities for continuous rotation servos; and the DC motors have individually set PWM duty cycles, adjusting their running power and speed. The DC motors have an independent power input and isolated ground to minimize crosstalk between DC motors and the controller logic on the board. For the servo motors and DC motors, the user can select a working PWM frequency for best performance depending on the motor used.

M-4B6SM-X1 Brief Spec:

- Number of Servo Motors: 6
- Max. Servo Motor Current: 5A per servo
- Servo Power Supply: 4V to 24V
- Min. and Max. Servo Duty Cycle – Per individual servo: 16bit, µs step size.
- Servo Working frequency – Global setting for all servos: 50/60/75/100/150/200/250/300 Hz.
- Servo Motor Resolution: 2048 steps [10bit] from Min. to Max duty cycle.
- Number of DC Motor Drivers: 4
- DC Motor Driver Type: 4 x independent, HEXFET configuration drivers.
- DC Motor Control: 256 steps [8bit] PWM.
- PWM frequency – Global setting for all DC Motors: 5kHz to 100kHz in steps of 2.5kHz.
- DC Motor Power Supply: 4V to 26V
- Max. DC Motor Current: 20A peak, 10A continuous per motor*
- Motor Current Draw in Disabled State: ≈ 0mA
- Controller Logic: FPGA based.
- Controller Logic Power Supply: 6V to 15V
- Controller Logic, Current: 25-50mA
- Board Interface: USB2RS232 bridge (USB 2.0 type B) or 1.2V to 5V direct RS232 interface (3 pin header, jumper selectable)
- Board Dimensions: 6.30in x 3.94in x 0.75in (160mm x 100mm x 19mm)

* Current rating is for 30°C ambient temperature in open air. Board performance may be reduced If it is used in tightly enclosed areas or at high ambient temperatures unless additional cooling is provided.

M-4B6SM-X1 Overview:

Connections and Board Layout:

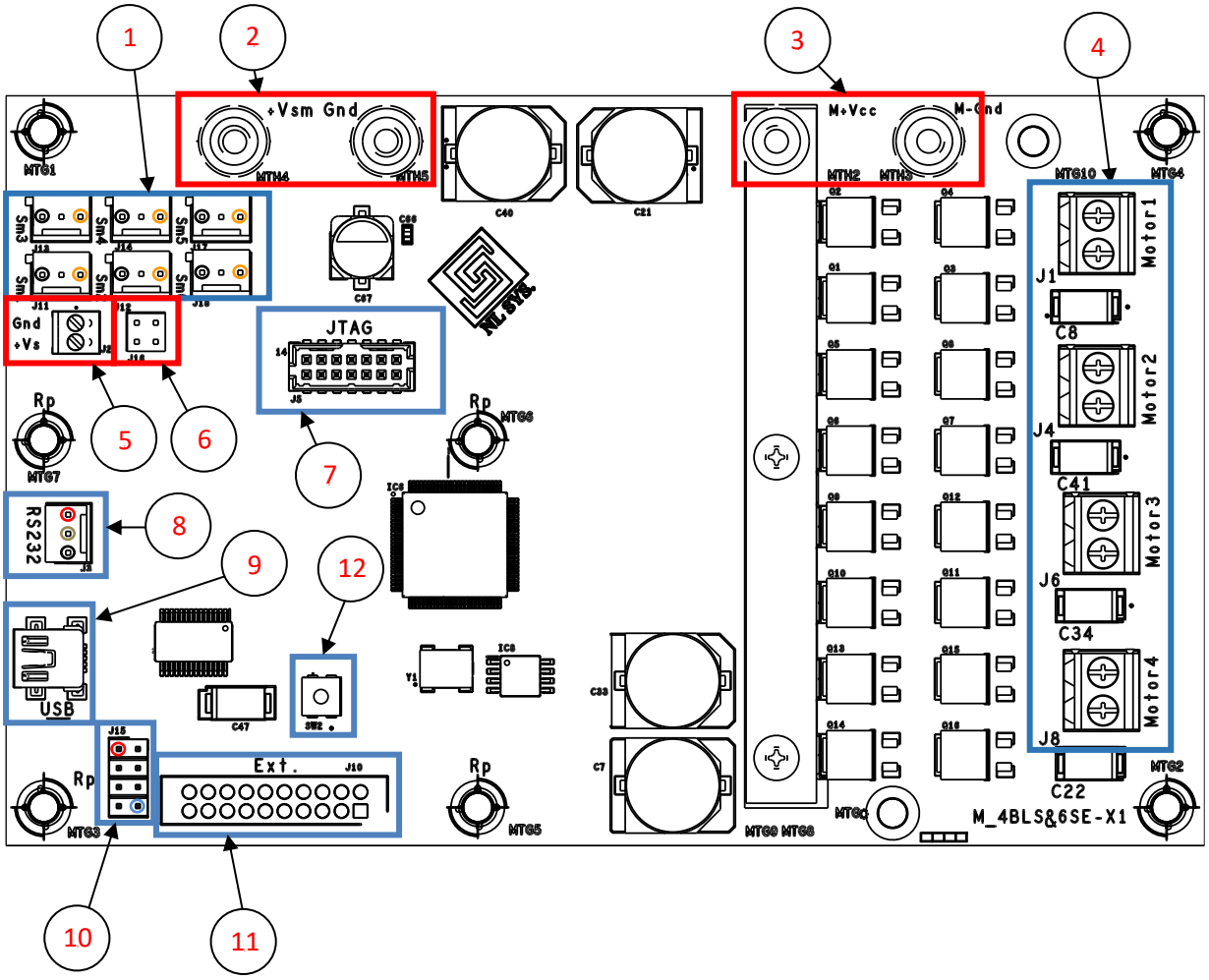
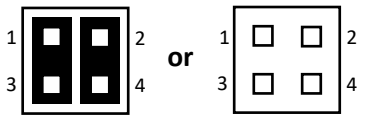


Figure 1: M-4B6SM Board Layout.

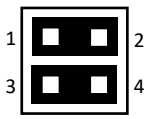
Description:

1. Servo motors 3 pin headers: J11 [Servo Motor 1], J12[Servo Motor 2], J13[Servo Motor 3], J14[Servo Motor 4], J17[Servo Motor 5], J18[Servo Motor 6].
   - Pin 1 → GND (Black)
   - Pin 2 → Servo motor Power Supply
   - Pin 3 → Servo Motor Pulse (Orange)
2. Servo motors power supply: MTH4(+), MTH5(-) → Voltage: 4V – 25V, Current: 5A per motor.
3. Brushed motors power supply: MTH2(+), MTH3(-) → Voltage: 4V – 26V, Current: Peek 20A per DC motor, Continuous 10A per DC motor.
4. DC motor connection terminal: J1[M1], J4[M2], J6[M3], J8[M4].

5. Board logic power supply: J2 → Voltage: 6V – 15V. Current: < 50 mA.
6. Power supply jumper: J16 → Selects power supply for board logic. Default setting is pins 1 & 2, and pins 3 & pin4 connected, or no pins connected. With default setting, power supply for logic and servo motors are separate and must be provided externally. If the selected servo motors power supply is in the range 6V – 15V, the servo power can be shared with the board logic by connecting pins 1 & 3 and pins 2 & 4**.**
   Important! – "Under no circumstances, should the motor power supply be connected to the servo motor and logic power supply without appropriate buffer (voltage regulator or similar). Connecting these two power supplies without the buffer can permanently damage the board!"

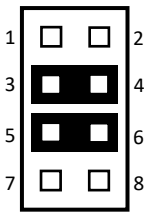


Independent Logic and Servo Power

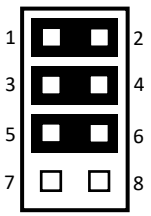


Shared Logic and Servo Power

7. FPGA JTAG connector: Standard XILINX Platform Cable USB II connector.
8. RS232 direct low voltage input: J3 → Serial voltage 1.2V – 5V. Must be selected on the mode selector header J15 (Item 10).
9. USB connector: USB2RS232 input → Default communication port. It can be switched by the mode selector header J15 (Item 10).
10. Mode selector: J15.
    Pins → 1&2 serial input select, Open = USB2RS232, Closed = direct RS232.
    Pins 3&4, 5&6, 7&8 → FPGA configuration mode M1/M2/M0 (Does not need to be changed for standard operation).



USB2RS232 Selected



Direct RS232 Selected

11. Test pins or expansion board: J10. Pin 1 is in the lower right of the connector marked by the square on the PCB. Odd numbered pins are on lower row and even numbered pins are on the upper row; both increasing from right to left.
    Pins 1/2/3/5/7/9/11/13/15/17/19/20 → GND
    Pin 4 → [D0] [CLK 1MHz – internal reference clock]
    Pin 6 → [D1] [SM1 – Servo1 motor pulse]
    Pin 8 → [D2] [SM2 – Servo2 motor pulse]

Pin 10 → [D3] [PWM Reference pulse]
Pin 12 → [D4] [Motor1 CCW_PWM pulse]
Pin 14 → [D5] [Motor1 CW_PWM pulse]
Pin 16 → [D6] [Motor4 CCW_PWM pulse]
Pin 18 → [D7] [Motor4 CW_PWM pulse].

12. FPGA reload button: SW2 → FPGA will reload image and reset all registers inside.

List of Registers:

The M-4B6SM-X1 board is controlled and configured through a series of 16-bit registers. Some functions, use the full 16-bits, while others use the most and least significant 8-bits for different functions. To write to a register, send the write code 0x01 followed by the 8-bit register address and finally the 16-bits of data. To read from a register, send the read code 0x02, followed by the register address; the board will respond with the 16-bits stored in the register. Table 1 lists all available registers.

Table 1: List of Registers

Register:	**Address: (Hex)**	**Data: 16-bits**		**Description:**
		Most Sig. 8-bits	**Least Sig. 8-bits**	
enableDevice	0x00	Enable Code – (0=Disable, 1=Enable) [NC, NC, NC, NC, M4, M3, M2, M1, NC, NC, S6, S5, S4, S3, S2, S1]		Enables or Disables selected motors or servos. (0x0000 disables all, 0x0f3f enables all)
revCode	0x01	16-bit Revision Code (Read Only)		FPGA firmware revision number
servoPos1	0x02	Servo 1 Position – (0 to 2048)		Sets servo 1 position between the min and max
servoMin1	0x03	Servo 1 Min – (500 default)		Sets the PWM duty cycle in µs for the min position
servoMax1	0x04	Servo 1 Max – (2500 default)		Sets the PWM duty cycle in µs for the max position
servoPos2	0x05	Servo 2 Position – (0 to 2048)		Sets servo position between the min and max
servoMin2	0x06	Servo 2 Min – (500 default)		Sets the PWM duty cycle in µs for the min position
servoMax2	0x07	Servo 2 Max – (2500 default)		Sets the PWM duty cycle in µs for the max position
servoPos3	0x08	Servo 3 Position – (0 to 2048)		Sets servo position between the min and max
servoMin3	0x09	Servo 3 Min – (500 default)		Sets the PWM duty cycle in µs for the min position
servoMax3	0x0a	Servo 3 Max – (2500 default)		Sets the PWM duty cycle in µs for the max position

servoPos4	0x0b	Servo 4 Position – (0 to 2048)		Sets servo position between the min and max
servoMin4	0x0c	Servo 4 Min – (500 default)		Sets the PWM duty cycle in µs for the min position
servoMax4	0x0d	Servo 4 Max – (2500 default)		Sets the PWM duty cycle in µs for the max position
servoPos5	0x0e	Servo 5 Position – (0 to 2048)		Sets servo position between the min and max
servoMin5	0x0f	Servo 5 Min – (500 default)		Sets the PWM duty cycle in µs for the min position
servoMax5	0x10	Servo 5 Max – (2500 default)		Sets the PWM duty cycle in µs for the max position
servoPos6	0x11	Servo 6 Position – (0 to 2048)		Sets servo position between the min and max
servoMin6	0x12	Servo 6 Min – (500 default)		Sets the PWM duty cycle in µs for the min position
servoMax6	0x13	Servo 6 Max – (2500 default)		Sets the PWM duty cycle in µs for the max position
motorSpeed1	0x14	Motor 1 Dir. - (0 or 1)	Motor 1 Speed – (0 - 255)	Sets motor direction and speed. Motor brake when motor is enabled and speed set to 0.
motorSpeed2	0x15	Motor 2 Dir. – (0 or 1)	Motor 2 Speed – (0 - 255)	Sets motor direction and speed. Motor brake when motor is enabled and speed set to 0.
motorSpeed3	0x16	Motor 3 Dir. – (0 or 1)	Motor 3 Speed – (0 - 255)	Sets motor direction and speed. Motor brake when motor is enabled and speed set to 0.
motorSpeed4	0x17	Motor 4 Dir. – (0 or 1)	Motor 4 Speed – (0 - 255)	Sets motor direction and speed. Motor brake when motor is enabled and speed set to 0.
pwmFreq	0x18	Servo Drive Freq. 0 = 50Hz, 1 = 60Hz - Def, 2 = 75Hz, 3 = 100Hz, 4 = 150Hz, 5 = 200Hz, 6 = 250Hz, 7 = 300Hz	Motor Drive Freq. (0 – 31) Default = 6 Freq. = (5 + 2.5 * Set_Value) kHz	Sets the driver frequency for all servos and motors
measFreq	0x19	Measured Motor Drive Freq. (Read-Only)		The measured motor driving frequency in 10 Hz increments.

Using the M-4B65M:

Connecting the Battery or Power Supply

The board has three power supply inputs, which include: 4V-26V DC motor power, 4V-25V servo power and 6V-15V board logic power (Items 2, 3 and 5 in Figure 1). These power inputs can operate separately, or if the servo supply voltage is between 6V to 15V, it can be shared with the board logic. To share the servo with the board logic, connect pins 1 and 3, as well as pin 2 and 4 on jumper J16 (Item 6 in Figure 1). Examples with the power supplies separated and connected, are shown in Figures 2 and 3.

It is also possible to share the DC motor power input with the servo power input if an appropriate buffer is used, such as a voltage regulator. Care should be taken to select an appropriate buffer that is able to withstand the induced return voltage from the DC motors while they are changing speed. The induced return voltage can be significant, especially when using larger motors, systems with high mechanical inertia and performing rapid speed changes.

Warning! - Under no circumstances, should the motor power supply be connected to the servo motor and logic power supply without appropriate buffer (voltage regulator or similar). Connecting these two power supplies without the buffer can permanently damage the board!

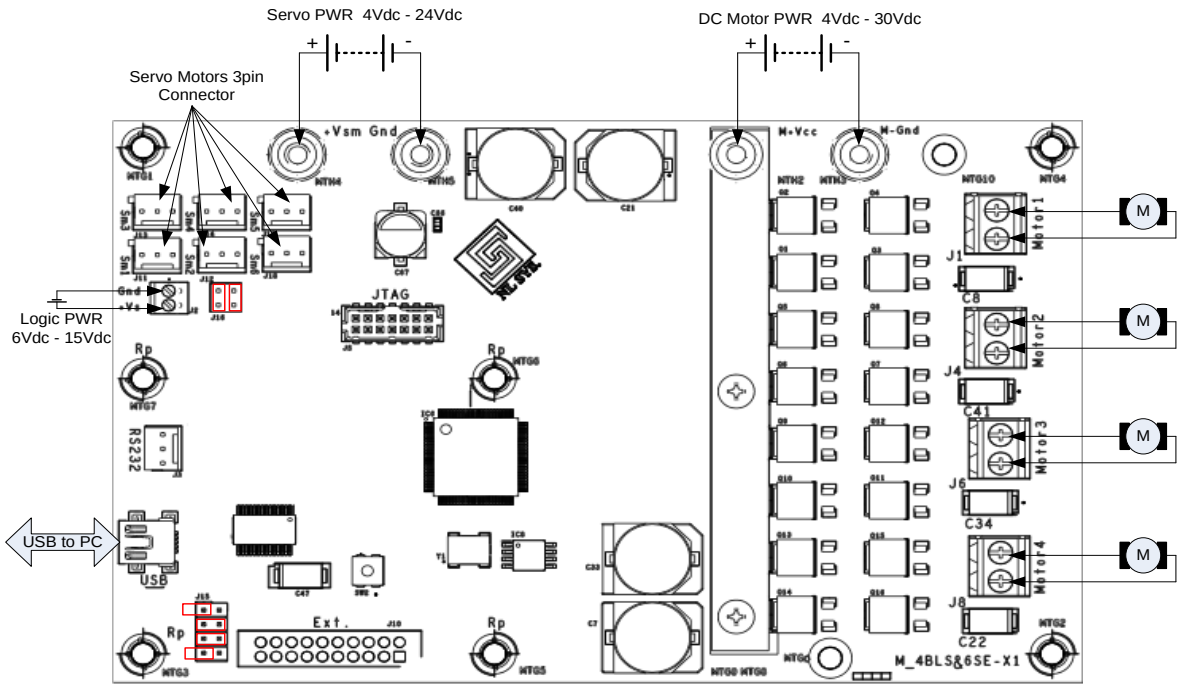


Figure 2: Connection with all 3 separate power supplies with USB2RS232 bridge

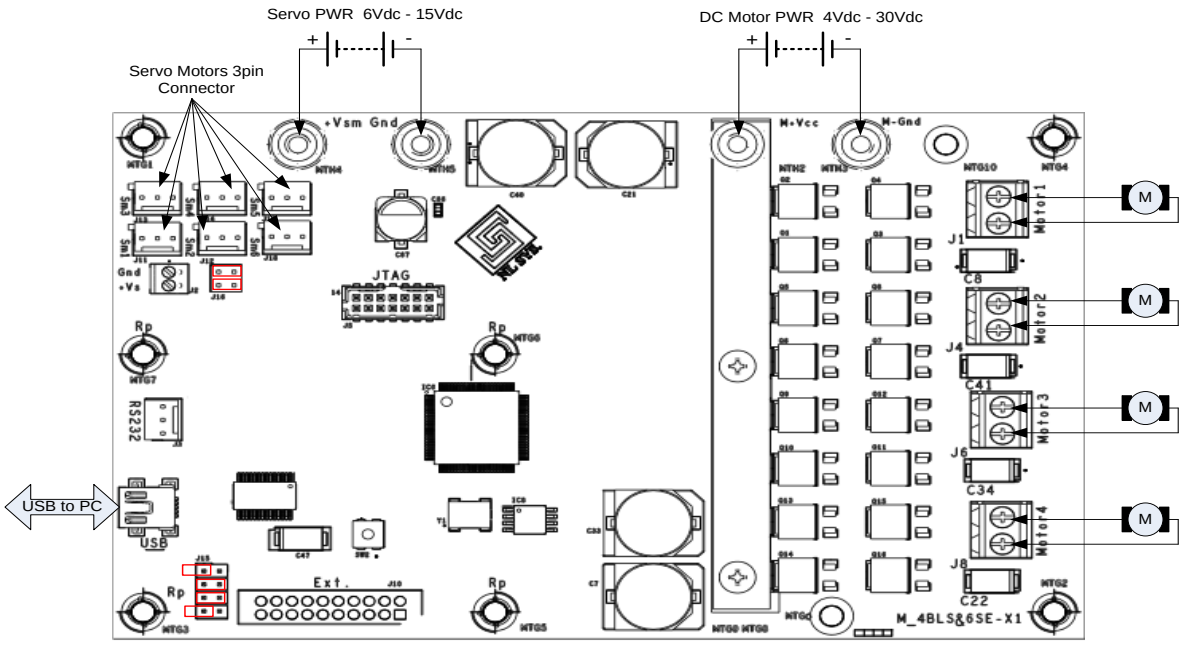


Figure 3: Connection with 2 power supplies using USB2RS232 bridge

Connecting the Motors

DC Motors are connected on the 4 terminal blocks on the edge of the board (Item 4 in Figure1). The polarity of the block can be switched with the direction setting on the board, so the motors should be wired to rotate in the desired forward and reverse directions. Servo motors can be connected to the 6 servo headers at the corner of the board (Item 1 in Figure 1)

Communicating with the Board

Communication with the board is performed by a serial connection, either using the integrated USB2RS232 connection (Item 9 in Figure 1) or over a low voltage, 1.2V to 5V, RS232 connection (Item 8 in Figure 1). Jumper J15 (Item 10 in Figure 1) selects which communication method is used. Figures 2 and 3 show the USB2RS232 configuration and Figure 4 shows the low voltage RS232 configuration.

To write a command to the board, a write select byte, 0x01, must be sent first, followed by the address of the register to write to and finally two bytes of data to store in the register. The register value will be kept until the board is powered off, or the FPGA Reload button is pressed (Item 12 in Figure 1).

Similarly, to read from the board, the read select byte must be sent, 0x02, followed by the address of the register to be read. After the board receives the address of the register, it will send back the two bytes of data stored in the register.

Enabling and Running DC Motors

To run any DC motors connected to the board, the desired motor port must be enabled, and the direction and speed must be set. If the motor speed is set before being enabled, the motor will start running at the set speed as soon as it is enabled. The board will default to a speed setting of 0 when

powered on, but it will retain the speed setting if the motor is disabled without powering down or resetting the board. To ensure the motor does not run immediately after being enable, a speed setting of 0 should be loaded before it is enabled.

To enable motors, an enable code must be written to the enableDevice register. The enable code is binary toggle for each connected device. The binary toggle pattern is: [NC, NC, NC, NC, M4, M3, M2, M1, NC, NC, S6, S5, S4, S3, S2, S1], where NC=No Connect, M=Motor and S=Servo. Toggling the NC positions will not affect the function of the other devices. Therefore, to enable all devices the register should be set to 0x0f3f or 0xffff, and to disable all devices the register should be set to 0x0000. The current enable code can be checked by reading from the enableDevice register.

The register for the desired motor port must be loaded with the direction and speed. Table 1 contains a list of all available registers. The direction can be set to 0x00 for forward or 0x01 for reverse and the desired motor speed that can range from 0 to 255. The speed and direction can be updated by writing to the register again. The board is capable of reversing direction immediately, however this will apply undue stress to the board and motors. It is therefore recommended to set the speed to 0 and allow the motor to slow down before setting a speed in the opposite direction,

The current speed and direction setting can be checked by reading from the register for the desired motor port. This register will update even with the port disabled and can be used to confirm a working serial connection without running any motors.

Example:

Task: Motors 1 and 2 need to be set to full power in the reverse direction and motors and motors 3 and 4 need to be set to half power in the forward direction. The setting for Motor1 needs to be confirmed after being sent.

Solution: First the enable code needs to be loaded into the board. Since we are running all the motors, all the “M” positions need to be set to 1. This results in a code of 0x0f00 in hexadecimal. Next it should be written to the enableDevice register by sending the write command 0x01, followed by the address 0x00, and then the two enable code bytes: 0x0f and 0x00. Now all the motor ports are enabled while all the servo ports remain disabled.

After the motors are enabled, the speed and direction need to be set. For Motor1, this can be done by sending the write command 0x01, followed by the address for Motor1: 0x14, then the direction 0x01 and finally the speed 0xff. For Motor2 the procedure will be the same, but with the address 0x15 instead of 0x14, resulting in the following bytes: 0x01, 0x15, 0x01, 0xff.

The procedure for motors 3 and 4 will be similar, with the addresses changed to 0x16 and 0x17, direction to 0x00 and the speed setting being changed to 127, which is 0x7f in hexadecimal. This will result in sending bytes: 0x01, 0x16, 0x00, 0x7f for Motor3 and 0x01, 0x17, 0x00, 0x7f for Motor 4.

The current setting of the Motor1 connection can be checked by sending the read code 0x02, followed by the address 0x14 and then reading the next two bytes sent from the board. The first byte will be the motor direction and the second byte will be the current speed setting for the motor.

Motor Braking and Freewheeling

The DC motor drivers use a power-break PWM system that will provide breaking if the motor is driven faster than its natural speed at the set PWM duty cycle. This allows the motor to provide break force if its speed setting is lowered until it reaches its new natural speed. The greater the difference between the motors current speed and its set speed, the greater the provided break force.

Full motor brake can be applied by setting the motor speed for the desired brake motor to 0 while the motor is enabled. The motor will continue to brake until the speed setting is changed or the motor is disabled in the enableDevice register. For high inertia systems, care should be taken not to overheat the motor during heavy prolonged braking since all the energy is dissipated through the braking motor.

To allow a motor to freewheel and no longer provide power or drag to the system, it must be disabled in the enableDevice register. This is the only state where the motor will be allowed to freewheel.

Controlling Servo Motors

The servo motors are controlled by sending the enable to code to enableDevice register and updating the servo's servoPos register. Unlike the DC Motors, servo motors have a minimum and maximum position duty cycle, which are given by the servo supplier. If the values are different from the default 500µs and 2500µs, min. and max. values, the correct values need to be loaded into the servoMin and servoMax registers for the servo to function properly.

A servo with an improperly configured min and max will either have a reduced angle range or vibrate when set to a min or max position. If either of these cases occur, the min and max values should be adjusted until the full range of the servo is available without vibration at the extremes.

Before enabling a servo, the servoMin, servoMax, registers should be updated, if they are different from defaults. This is done to prevent unwanted servo movement and vibration when enabled. The min and max values are in microseconds and stored in 16-bit registers. A register value of 0x898 would correspond to a setting of 2200µs. The initial servoPos can be updated as well if the initial position of the servo is a concern. The servo position will always have a value between 0 and 2048, with 0 being the position set by servo min, and 2048 being the position set by servo max. All other values being normalized between the extremes.

To set the initial min and max, first send the write byte command 0x01, followed by the address of the servoMin or servoMax register for the servo port used, as listed in Table 1. These registers use the full 16-bits to set the position, so the first data byte sent will be the most significant 8-bits, followed by the least significant 8-bits of the value in microseconds.

As with the DC Motors, to enable the servo, the proper enable code must be loaded into the enableDevices register. The enable code is binary toggle for each connected device. The binary toggle pattern is: [NC, NC, NC, NC, M4, M3, M2, M1, NC, NC, S6, S5, S4, S3, S2, S1], where NC=No Connect, M=Motor and S=Servo. Toggling the NC positions will not affect the function of the other devices. Therefore, to enable all devices, the register should be set to 0x0f3f or 0xffff, and to disable all devices the register should be set to 0x0000. The current enable code can be checked by reading from the register.

Once the servo is enabled, the position can be updated at any time, the servo will move to the new position after the register is loaded. Due to the servo position always having 2048 steps, some servos may not be sensitive enough to use the full precision of the board and will only move every few steps.

Note: - While disabled, some servo motors will maintain their last set position, while others will allow free rotation.

The current settings for the servo can be checked by reading from the registers for the desired value. For the servo ports, all the registers are used for a single value and have 16-bits of data. To read from the register, first send the read command 0x02, followed by the address. The board will respond with the full 16-bits, MSB first, and can be read as two 8-bit bytes.

Example:

Task: Servo1 needs to start at 25% rotation, then move to 100% and back to 35%. The 35% position also needs to be confirmed. The servo has a minimum position of 700µs and a maximum of 2200µs.

Solution: To start, the min, max and initial position need to be loaded into their registers. The min position can be set by sending the write command, 0x01; then the address for the min setting for servo1, 0x03; followed by the 2 data bytes to set the register to 700 which are 0x02 and 0xbc. The max position can be set with the same procedure, but using the address 0x04 and a data value of 2200, which are 0x08 and 0x98 respectively in hex. This would result in the following bytes being sent: 0x01, 0x04, 0x08, and 0x98. The initial servo position needs to be set to 25%, since the position register is always 0 to 2048, this will result in a setting 512. To apply this setting the following bytes would need to be sent: 0x01, 0x02, 0x02, 0x00.

Once the min, max and initial position are set, the servo can be enabled. For this example, only the servo being used will be enabled. Toggling the “S1” position in the enable code gives us 0x0001. The enable code can then be written to the register using: 0x01, 0x00, 0x00, 0x01. The servo should now be energized and maintain the set position.

Further changes to the servo can be written to the register when needed. To move to the 100% position, the following bytes need to be sent: 0x01, 0x02, 0x08, 0x00. To move the 35% position, the closest step increment needs to be calculated: 2048*0.35 = 716.8. After rounding, the closest setting is

717 or 0x02cd in hex. The new setting can be applied by sending the following bytes: 0x01, 0x02, 0x02, 0xcd.

The 35% setting can be confirmed by reading from the position register for Servo1. This can be done by sending the read command 0x02, followed the Servo1 address 0x02 and reading the next two bytes sent by the board. The two bytes should match the set value of 717 or 0x02 0xcd in hex.

Setting the PWM Frequency

The pwm driver frequency for the DC motors as well as the servos can be configured by writing into the pwmFreq register. The servo drive frequency is set by the first 8-bits of the register and can be set to values from 0 to 8. This corresponds to frequencies of: [50, 60, 75, 100, 150, 200, 250, 300] Hz. The default value is 1 with a frequency of 60Hz, The DC motor drive frequency is set by the second byte in the register and can be set from 0 to 31. The DC driver frequency is calculated by: Frequency = (5 + 2.5*N) kHz, where N is the set value in the register. The default value of N is 6 with a frequency of 20kHz. Generally, the larger the motor the lower the drive frequency needs to be for the best efficiency. However, for frequencies below 20kHz, the driver may make audible tones while running.

The frequency setting can be confirmed by reading from the pwmFreq register and for the DC motor drive frequency and can be internally measured by reading from the measFreq register. The measFreq register is read only and measures the actual drive frequency for the DC motor drivers in 10s of Hz. Therefore, to calculate the frequency in Hz, simply multiply the value from measFreq by 10. This value should match the calculated frequency setting for the DC motor driver from pwmFreq. A discrepancy in values between measFreq and pwmFreq may indicate a fault with the board.

Example:

Task: The servo drive frequency needs to be changed to 100Hz and the DC motor frequency needs to be set to 35kHz. The setting needs to be confirmed by both reading from the setting register and measuring the DC motor drive frequency directly.

Solution: To set the servo drive frequency, the first byte of the register needs to be set to a value corresponding to the desired frequency. From the list of frequencies for the register in Table 1, 100Hz corresponds to a set value of 3. For the DC motor driver frequency, the second byte value needs to be calculated to match the desired frequency. From the equation for the register in Table 1, the byte value should 12 for a drive frequency of 35kHz. To write the new values to the register, first the write command is sent 0x01, followed by the register address 0x18, then the servo frequency setting 0x03, followed by the DC motor frequency setting 0x0c.

To check the frequency settings, first send the read command 0x02, followed by the address 0x18. The first byte received is the current servo frequency setting and the second byte is the current DC motor frequency setting. If the register was updated, the bytes should match the values sent earlier of 0x03 and 0x0c.

To confirm the actual DC motor drive frequency matches the setting, the measFreq register can be read by sending read command 0x02, followed by the address 0x19. The next two bytes received are the internally measured drive frequency in 10Hz increments. To calculate the drive frequency in Hz, multiply the value from the register by 10. In this case the register should read 0x0dac in hex, which when converted to decimal and multiplied by 10Hz, is 35,000Hz and matches the set frequency.

Additional Board Information

A block diagram for the entire board is shown in Figure 4, and the FPGA pinouts are given in Table 2. The board features a SPARTAN XC3S200A FPGA with the Voltages: VCCO = 3.3V, VCCAUX = 3.3V, VCCINT = 1.2V. Pin 1 of the FPGA is labeled with a white dot on the PCB.

This information is given to better understand the functions of the board and help with troubleshooting. The board and FPGA code should not be modified, as it may result in damage to the board, or other components attached to the board.

Warning! - Any changes to the FPGA code or to the board itself may results in damage to the board or other connected devices.

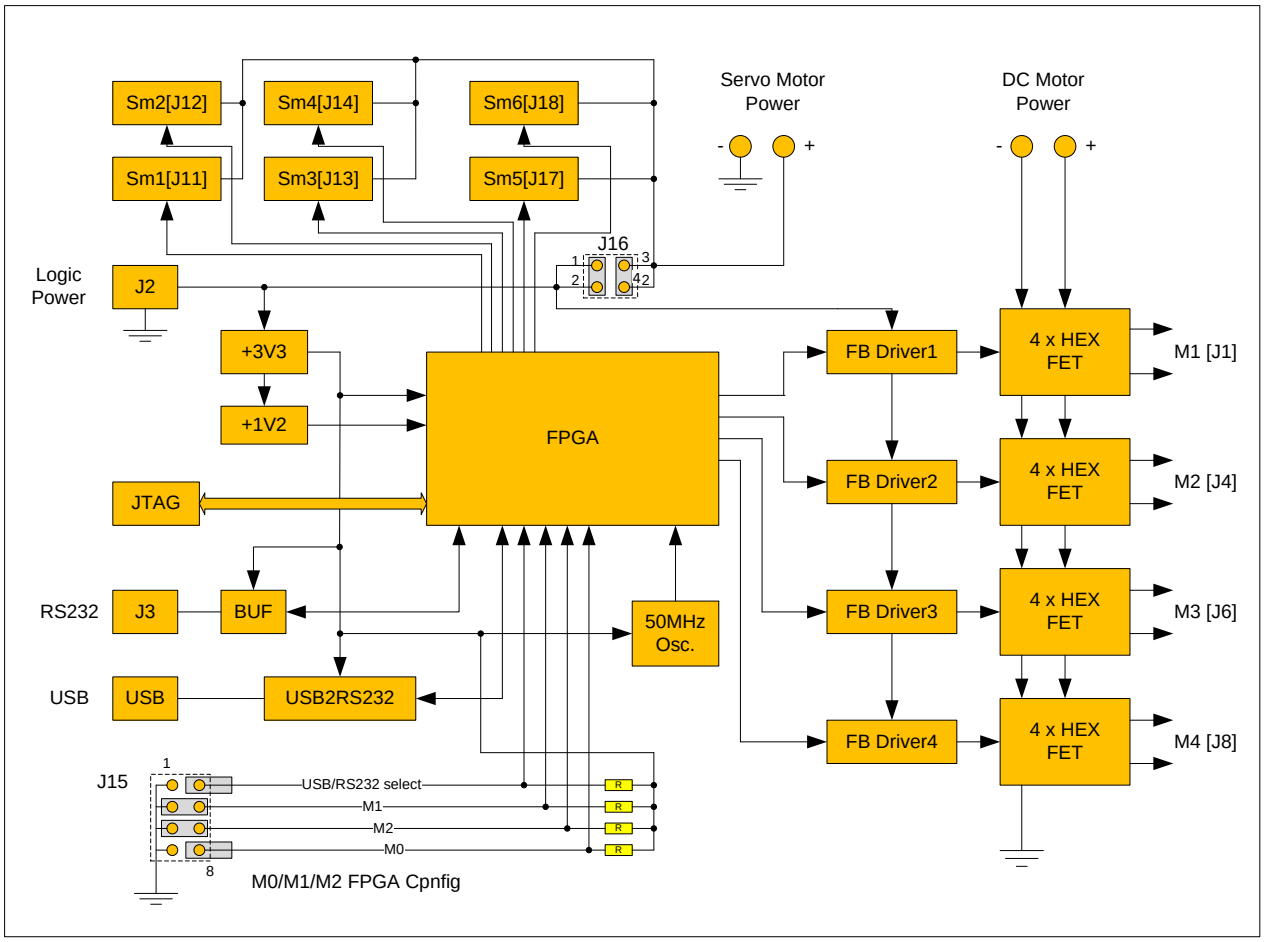


Figure 4: M-4B6SM-X1 simplified block diagram with default jumper position J16 (Servo and logic power) and J15 (USB or RS232 Select).

Table 2: FPGA Pin Assignments.

Names	Pin Location	Function
"TXD"	LOC = "P3";	# | input
"NDTR"	LOC = "P4";	# | NC
"NRTS"	LOC = "P5";	# | NC
"RXD"	LOC = "P6";	# | output
"NDSR"	LOC = "P9";	# | NC
"NDCD"	LOC = "P10";	# | NC
"NCTS"	LOC = "P12";	# | NC
"NR"	LOC = "P13";	# | NC
"NPWR"	LOC = "P15";	# | NC
"TRST"	LOC = "P16";	# | NC
"NSLEEP"	LOC = "P19";	# | NC
"BUSDETECT"	LOC = "P20";	# | NC
"INT_EXT_RS"	LOC = "P21";	# Select between USB2RS IC. J15 Pins Open = USB2RS232 J15 Pins Closed = RS232
"D [7]"	LOC = "P28";	# Test Pin
"D [6]"	LOC = "P29";	# Test Pin
"D [5]"	LOC = "P30";	# Test Pin
"D [4]"	LOC = "P31";	# Test Pin
"D [3]"	LOC = "P32";	# Test Pin
"D [2]"	LOC = "P33";	# Test Pin
"D [1]"	LOC = "P34";	# Test Pin
"D [0]"	LOC = "P35";	# Test Pin
"SM5"	LOC = "P37";	# Servo Motor 5 output
"SM6"	LOC = "P36";	# Servo Motor 6 output
"CLK1"	LOC = "P43";	# Spare Clock Input
"CLK0"	LOC = "P44";	# 50MHz clock input
"EN_1"	LOC = "P73";	# Enable Motor 1 output
"BPWM_1"	LOC = "P72";	# Motor 1 PWM CW output
"APWM_1"	LOC = "P71";	# Motor 1 PWM CCW output
"EN_2"	LOC = "P70";	# Enable Motor 2 output
"BPWM_2"	LOC = "P65";	# Motor 2 PWM CW output
"APWM_2"	LOC = "P64";	# Motor 2 PWM CCW output
"EN_3"	LOC = "P62";	# Enable Motor 3 output
"BPWM_3"	LOC = "P61";	# Motor 3 PWM CW output
"APWM_3"	LOC = "P60";	# Motor 3 PWM CCW output
"EN_4"	LOC = "P59";	# Enable Motor 4 output
"BPWM_4"	LOC = "P57";	# Motor 4 PWM CW output
"APWM_4"	LOC = "P56";	# Motor 4 PWM CCW output
"EXTRS232_TX"	LOC = "P94";	# Serial Output J3 [1]
"EXTRS232_RX"	LOC = "P93";	# Serial Input J3 [3]
"SM1"	LOC = "P89";	# Servo Motor 1 output
"SM2"	LOC = "P88";	# Servo Motor 2 output
"SM3"	LOC = "P86";	# Servo Motor 3 output
"SM4"	LOC = "P85";	# Servo Motor 4 output