MK-200C

User's Manual

Revision 1.1

MOTKA LLP

Table of Contents

List of Figures

List of Tables

1. Introduction

This User Manual targets application developers. It provides complete information on how to use the MK-200C.

For information on programming and the complete list of command references, please refer to the Command Reference manual.

MK-200C is a 2-Axis Motion Controller that facilitates implementation of motion control applications with simplicity, shortens time-to-market, and achieves optimal cost-effectiveness. It is suitable for scientific, industrial automation, robotic applications and hobby.

MK-200C is a fully functional motion controller requiring only an external amplifier to complete a position control application. It is driven by a host through an asynchronous serial port (RS-232C). Figure 1 shows the elements of motion control application using MK-200C. Its servo compensation uses 32-bit position error, as well as PID control engine with acceleration and velocity limits for position control. A set of essential and simple-to-use instructions is provided to control the motion application and monitor ongoing performance.

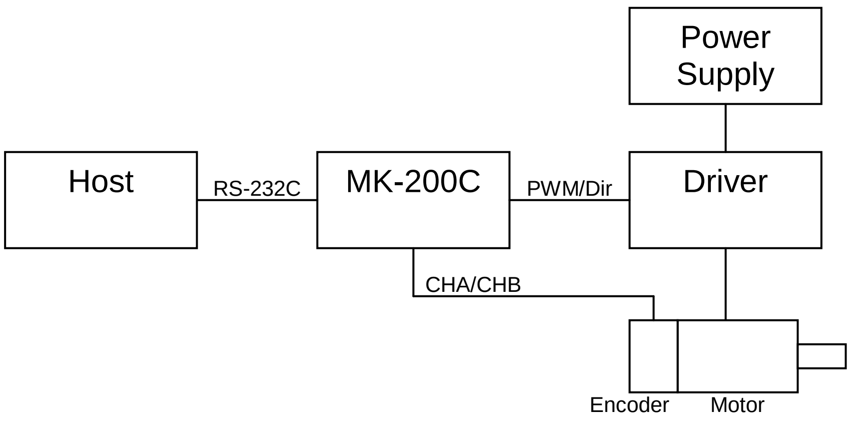

Figure 1 – Elements of motion control

Features

- Supports up to 2 axes
- Configurable to support step or servo motors
- 5V tolerant PWM/Pulse and Direction outputs per axis to ensure compatible with commercially-of-the-shelf amplifiers
- Two channels (CHA and CHB) incremental encoder quadrature input per axis
- Two directional (Forward and Reverse) limits per axis
- One home indicator per axis
- One RS-232C interface port, configured at 115200 bps, to interface with a host computer
- Easy-to-use ASCII-based programming instructions
- Small footprint (116.2 x 62.9 x 24.18 mm) with screw terminals for ease of integration and maintenance
- Lightweight with only 150 gram
- +5 V operation and typical current consumption of 300 mA.

Figure 2 – The MK-200C 2-Axis Motion Controller

Related documents

- Command Reference.
- MOTKA Motion Companion User’s Manual.

2. Getting Started

The terminal layout and definitions of MK-200C are shown in Figure 3 and Table 1 respectively.

The instructions illustrated from Section 2.1 through Section 2.8 represent a typical application.

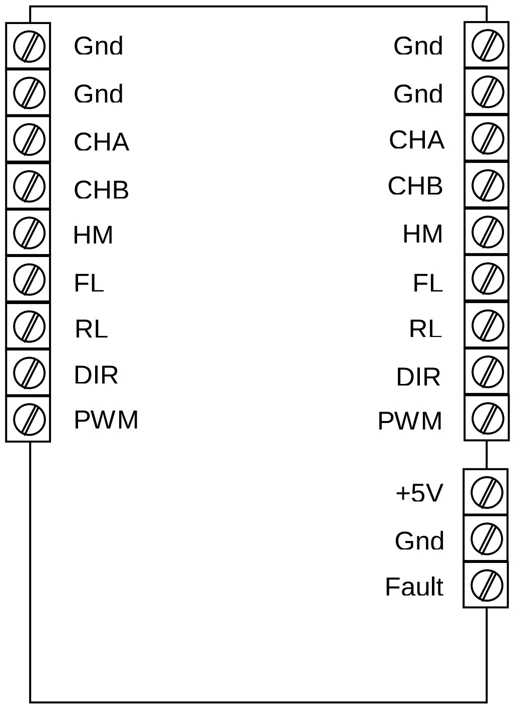

Figure 3 - MK-200C

Group	Terminal Name	Type[1]	I/O Level[2][3]	Function
AXIS A	Gnd	P	-	Ground
	CHA	I	FT	Axis-A Encoder input CHA
	CHB	I	FT	Axis-A Encode input CHB
	HM	I	FT,AL	Axis-A Home indicator input
	FL	I	FT,AL	Axis-A Forward limit input
	RL	I	FT,AL	Axis-A Reverse limit input
	DIR	O	FT	Axis-A Direction output
	PWM	O	FT	Axis-A PWM output
AXIS B	Gnd	P	-	Ground
	CHA	I	FT	Axis-B Encoder input CHA
	CHB	I	FT	Axis-B Encode input CHB
	HM	I	FT,AL	Axis-B Home indicator input
	FL	I	FT,AL	Axis-B Forward limit input
	RL	I	FT,AL	Axis-B Reverse limit input
	DIR	O	FT	Axis-B Direction output
	PWM	O	FT	Axis-B PWM output
Stepper Select	AXIS A	I	FT,AL	Axis A Stepper select
	AXIS B	I	FT,AL	Axis B Stepper select
Power In	+5V	P	-	+5V supply
	Gnd	P	-	Ground
	Fault	I	FT,AL	Reserved. Do not use.
RS-232	Rx	-	-	Receive
	Tx	-	-	Transmit
	Gnd	P	-	Ground

Table 1 – MK-200C terminal definitions

[1] P = Power, I = Input, O = Output
[2] FT = Five voltage Tolerant
[3] AL = Active Low

2.1. Connecting the Power Supply

The MK-200C requires a +5 V operating voltage. It is highly recommended to separate the power supply used for the MK-200C and the motor amplifiers to eliminate noise interference.

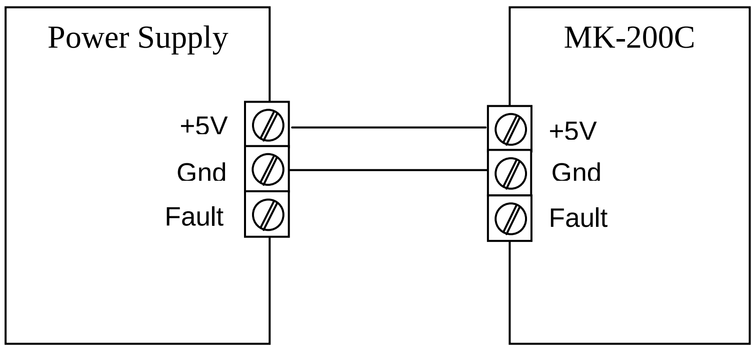

Figure 4 – Power supply scheme

2.2. Connecting the Host

A host is required to drive the MK-200C through the RS-232C interface with the following configurations:

Baud rate:	115200 bps
Data bit:	8
Stop bit:	1
Parity bit:	None
Flow control:	None

These configurations are not changeable.

The RS-232C interface can be connected directly to a host computer with a RS-232C port (See Figure 5).

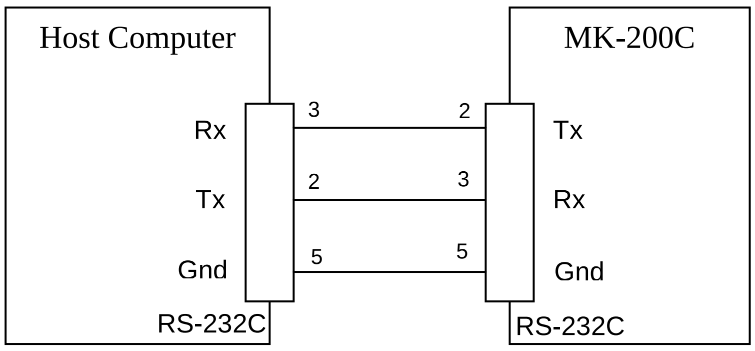

Figure 5 – RS-232C-to-RS232C interface with host

2.3. Selecting Servo or Stepper

The MK-200C supports 2 axes, namely Axis A and Axis B. Each axis can be configured to operate with a servo or stepper motor, depending on the jumper, STEPPER SEL (Axis A and Axis B), setting. Inserting the jumper configures the Axis to support stepper motor.

Both axes of MK-200 are, by default, configured to operate with stepper motors with both jumpers inserted. If servo motors are used in an application, simply remove the respective jumper. **It is important to note that the MK-200C reads these inputs only once upon power up. Any change of configuration to these inputs will not be registered until the next power up.**

2.4. Connecting Servo Motor

MK-200C supports standard dc servo motor amplifiers operate with PWM and Direction mode. It also supports standard 2-channel encoders with CHA and CHB outputs. All PWM and Dir outputs, as well as CHA and CHB inputs are 5 V tolerant. If amplifiers and encoders used in an application utilise other operating voltage other than +5 V, level-shifter must be used to match these voltages.

Figure 6 and Figure 7 show the typical connections of a brushless and brushed servo motor to an MK-200C's axis respectively. The jumper of STEPPER SEL for the respective axis must be removed.

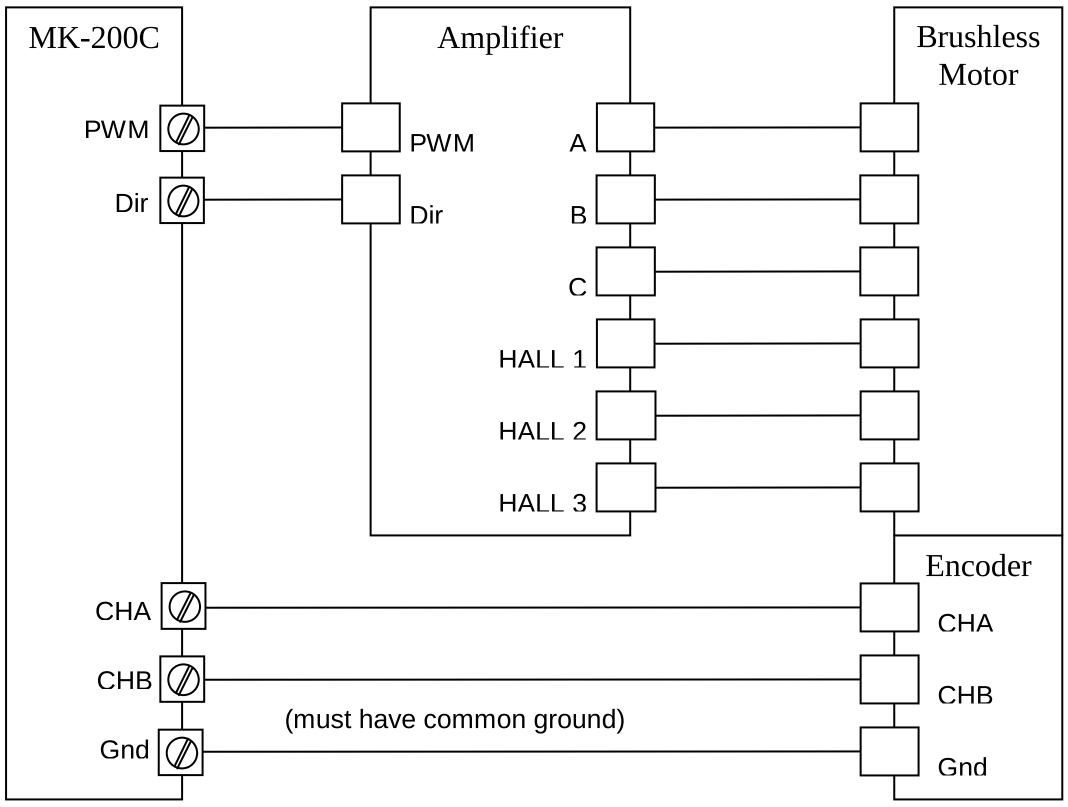

Figure 6 – Connecting to a brushless servo motor to MK-200C

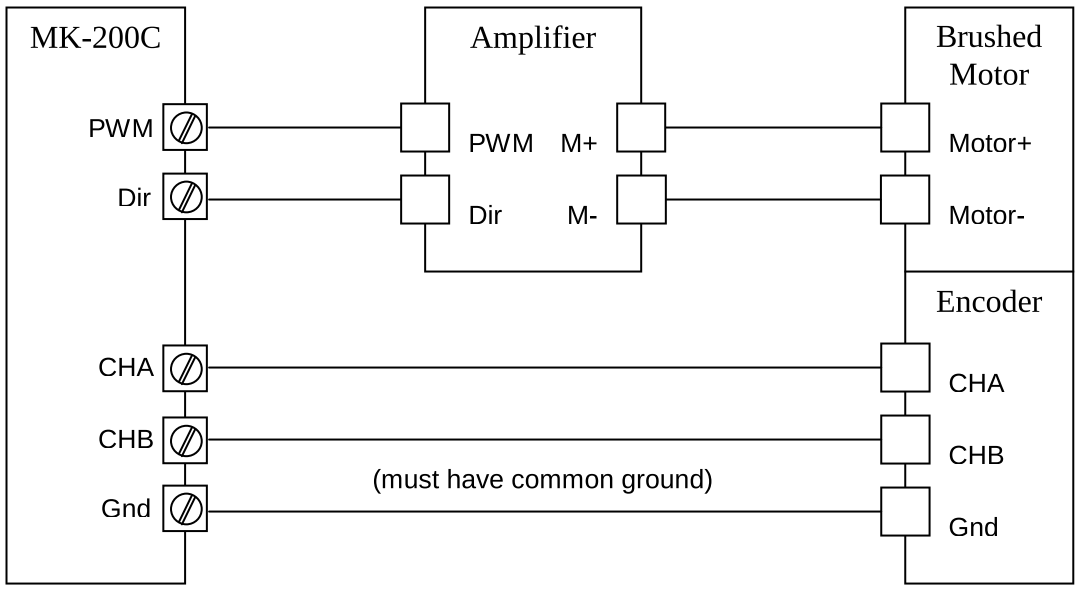

Figure 7 – Connecting to a brushed servo motor to MK-200C

2.5. Connecting Stepper Motor

MK-200C supports standard stepper motor amplifiers operate with PWM and Direction mode. **When stepper operation is selected, MK-200 does not support encoder inputs (CHA and CHB).**

Figure 8 shows the typical connection of stepper motor to an MK-200C's axis. The jumper of STEPPER SEL for the respective axis must be inserted.

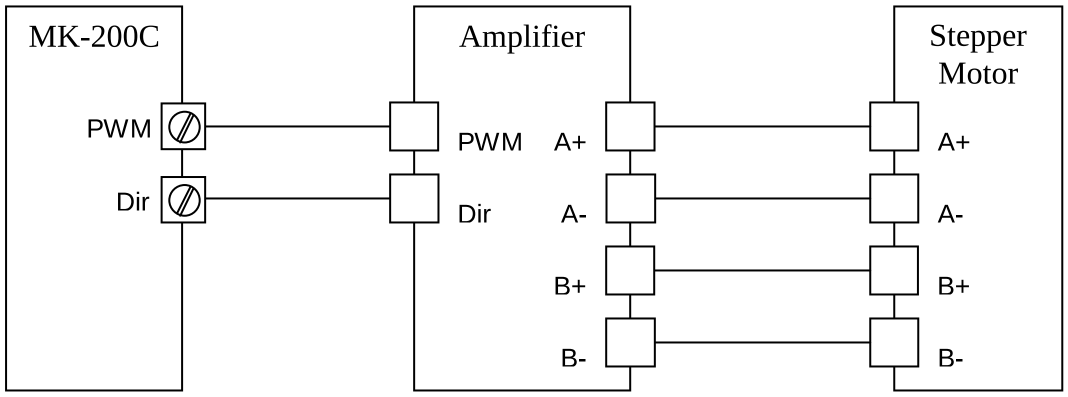

Figure 8 – Connecting to a stepper motor to MK-200C

2.6. Connecting the Limits

There are two directional limits per axis, which can be used to prevent collision due to over travelling. Leave these limits unconnected (inactive) if we do not wish to use them.

Both Forward and Reverse limits are active low.

If the Forward Limit (FL) is active, it inhibits the forward motion immediately. If the Reverse Limit (RL) is active, it inhibits the reverse motion immediately. After a limit has been activated, further motion in the direction of the limit will not be possible until the state of the limit returns back to inactive state. This usually involves physically moving the mechanisms or moving the motor in the opposite direction via instruction set.

Figure 9 depicts the connection of standard slot sensors to these limits.

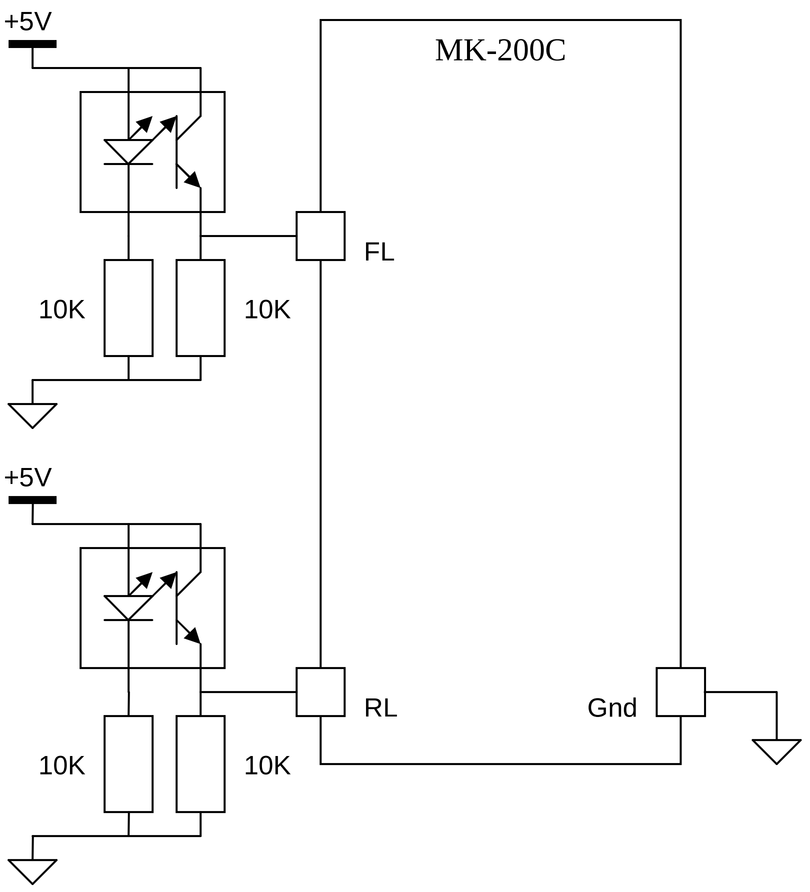

Figure 9 – Connecting to forward and reverse limits to MK-200C

2.7. Connecting the Home Indicator

The MK-200C comes with one Home indicator (HM) per axis. Home indicators are designed to provide mechanical reference points for a motion control application. A transition in the state of the home indicator alerts the controller that a reference point is reached by a moving part in a motion control system.

These inputs are active low and can be used with external sensors, such as slot sensor, as depicted in Figure 10.

Home operation is initiated by the Move Home (MH) instruction. MH instruction accepts both direction and speed of search. Refer to Command Reference for more information.

Move Home instruction initiates the corresponding PWM pulses and Direction bit. In the example shown Figure 11, the MH instruction searches in the forward (DIR is logic ‘0’) direction. Once the HM input is active (logic ‘0’), it toggles the Direction bit and reduces the speed of search, and hence reverses the motion at a lower speed. This motion continues until the HM input is inactive and the home operation is complete.

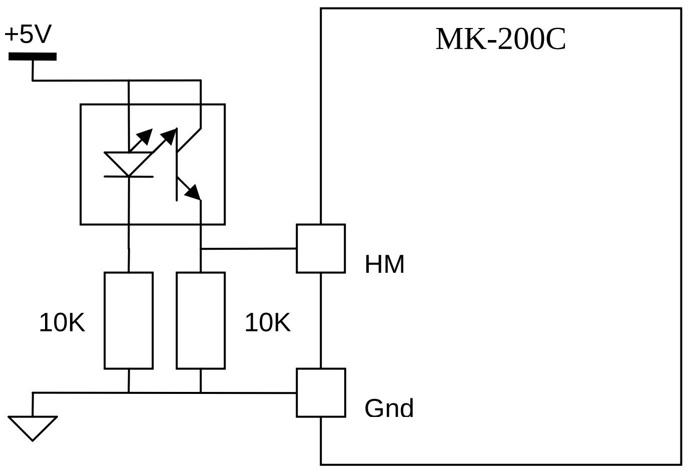

Figure 10 – Connecting to home indicator

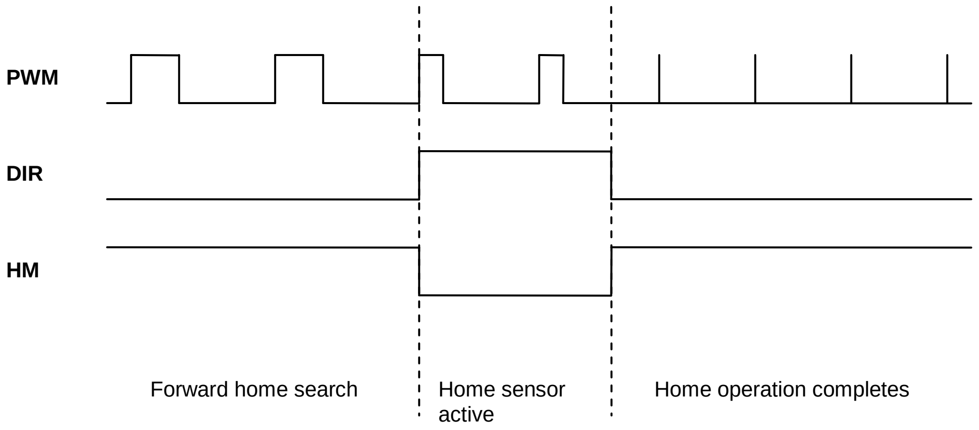

Figure 11 – Home operation

2.8. Testing Our Connections

Having connected all the necessary circuitry described in the above sections, we can start testing our application by sending a couple of instructions. To do that, we can use the MOTKA Motion Companion software or any third party software that can send and receive ASCII characters such as Microsoft HyperTerminal.

In the example below, we assume a servo motor is connected to Axis A. For more information to tailor your instructions to suit your tests, please refer to the Command Reference for complete set of instructions and explanations.

The first instruction is to check for Motor Type (MT). The response of '1' indicate that MK-200C is configured Axis A to operate in servo mode. Next we disable Acceleration Limit (AL) by setting it to zero. Before we can move the motor, it must be enabled by sending the Servo Enable (SE) instruction to Axis A. We get the first position (GP) and MK-200C responses with zero. We move the motor by 10,000 counts relative to current position. Depending on the tuning of PID parameters (KP, KI and KD) with respect to the loads, the response of the final position obtained by GP may be different from the example below.

```
MT ?
1
AL 0
SE T
GP ?
0
MR 10000
GP ?
10003
```

2.9. Design Examples

Here are a few simple C/C++ examples for using your controller. You may use other languages to send the ASCII commands via the serial port.

Example 1 – Tuning of Controller

This example assigns the PID parameters to Axis A and read back for confirmation.

```
CSerial port;
If(port.open(2, 115200))                    // Open serial port and configured
{                                           // to baud 115200 bps

        char Kp[7] = "KP 100";              // Set Proportional gain for Axis A
        Kp[6] = 0x0D;                       // Terminate it with a Carriage Return
        char Ki[7] = "KI 800";              // Set Integral gain for Axis A
        Ki[6] = 0x0D;                       // Terminate it with a Carriage Return
        char Kd[5] = "KD 2";                // Set Differential gain for Axis A
        Kd[4] = 0x0D;                       // Terminate it with a Carriage Return

        char query[5] = "KP ?";             // Query for KP
        query[4] = 0x0D;                    // Terminate it with a Carriage Return
        char readBuf[10];                   // Container for reply

        port.send(Kp, sizeof(Kp));          // Send KP to controller
        port.send(Ki, sizeof(Ki));          // Send KI to controller
        port.send(Kd, sizeof(Kd));          // Send KD to controller

        port.send(query, sizeof(query));            // Query for KP
        port.read(readBuf, sizeof(readBuf));        // KP is stored in readBuf
}
```

Example 2 – Profiled Move

In this example, Axis B moves a distance of 50,000 counts at the speed of 30,000 counts/sec and an acceleration and deceleration of 100,000 counts/sec^2. It motor stops once it reaches 50,000 counts.

```
CSerial port;
If(port.open(2, 115200))                    // Open serial port and configured
{                                           // to baud 115200 bps

        char SE[6] = "SE, T";               // Servo enable for Axis B
```

```
            SL[5] = 0x0D;                  // Terminate with Carriage Return
            char SL[10] = “SL, 30000”;     // Speed limit
            SL[9] = 0x0D;                  // Terminate with Carriage Return
            char AL[11] = “AL, 100000”;    // Acceleration limit
            AL[10] = 0x0D;                 // Terminate with Carriage Return
            char DL[11] = “DL, 100000”;    // Deceleration limit
            DL[10] = 0x0D;                 // Terminate with Carriage Return
            char MR[10] = “MR, 50000”;     // Move Relative
            MR[9] = 0x0D;                  // Terminate with Carriage Return

            port.send(SL, sizeof(SL));     // Set speed limit
            port.send(AL, sizeof(AL));     // Set acceleration limit
            port.send(DL, sizeof(DL));     // Set deceleration limit
            port.send(SE, sizeof(SE));     // Enable motor
            port.send(MR, sizeof(MR));     // Move motor
    }
```

Example 3 – Profiled Move of Multiple Axes

In this example, both Axis A and B move independently at the same time.

```
    CSerial port;
    If(port.open(2, 115200))                   // Open serial port and configured
    {                                          // to baud 115200 bps

            char SE[8] = “SE T, T”;                 // Servo enable for Axis A and B
            SL[7] = 0x0D;                           // Terminate with Carriage Return
            char SL[15] = “SL 30000, 5000”;         // Speed limits
            SL[14] = 0x0D;                          // Terminate with Carriage Return
            char AL[17] = “AL 100000, 80000”;       // Acceleration limits
            AL[16] = 0x0D;                          // Terminate with Carriage Return
            char DL[17] = “DL 100000, 10000”;       // Deceleration limits
            DL[16] = 0x0D;                          // Terminate with Carriage Return
            char MR[16] = “MR 50000, 75000”;        // Move Relative
            MR[15] = 0x0D;                          // Terminate with Carriage Return

            port.send(SL, sizeof(SL));              // Set speed limits
            port.send(AL, sizeof(AL));              // Set acceleration limits
            port.send(DL, sizeof(DL));              // Set deceleration limits
            port.send(SE, sizeof(SE));              // Enable motors
```

```
        port.send(MR, sizeof(MR));            // Move motors
}
```

Example 4 – Reading of Position

The position of any axis can be queried with the GP instruction.

```
CSerial port;
If(port.open(2, 115200))                  // Open serial port and configured
{                                         // to baud 115200 bps

        char query[7] = "GP ?,?";         // Get position – A and B axes
        query[6] = 0x0D;                  // Terminate with Carriage Return
        char readBuf[10];                 // Container for reply
        char posA[5];                     // Buffer for Axis A position
        char posB[5];                     // Buffer for Axis B position
        int i = 0;

        port.send(query, sizeof(query));          // Query for position

        // Once controller received the command, it will reply the positions.
        port.read(readBuf, sizeof(readBuf));      // Read the positions

        // Positions replied contained in readBuf are separated by 0x0D
        // Read position of Axis A
        while(readBuf[i] != 0x0D)
        {
                posA[i] = readBuf[i];
                i++;
        }
        // Read position of Axis B
        while(readBuf[i] != 0x0D)
        {
                posB[i] = readBuf[i];
                i++;
        }
}
```

Example 5 – Velocity Control

In this example, we drive A and B motors at a desired speed.

```
CSerial port;
If(port.open(2, 115200))                  // Open serial port and configured
{                                         // to baud 115200 bps

        char SE[8] = “SE T, T”;                   // Servo enable for Axis A and B
        SE[7] = 0x0D;                             // Terminate with Carriage Return
        char MC[15] = “MC 1000, -2000”;           // Speed and direction
        MC[14] = 0x0D;                            // Terminate with Carriage Return
        char AL[18] = “AL 100000, 300000”;        // Acceleration
        AL[17] = 0x0D;                            // Terminate with Carriage Return
        char DL[16] = “DL 50000, 50000”;          // deceleration
        DL[15] = 0x0D;                            // Terminate with Carriage Return
        char HT[8] = “HT T, T”;                   // Halt
        HT[7] = 0x0D;                             // Terminate with Carriage Return

        port.send(AL, sizeof(AL));                // Set acceleration limits
        port.send(DL, sizeof(DL));                // Set deceleration limits
        port.send(SE, sizeof(SE));                // Enable motors
        port.send(MC, sizeof(MC));                // Move motors

        wait(5000);                               // Wait for 5 seconds
        port.send(HT, sizeof(HT));                // Halt motors
}
```

3. Package Information

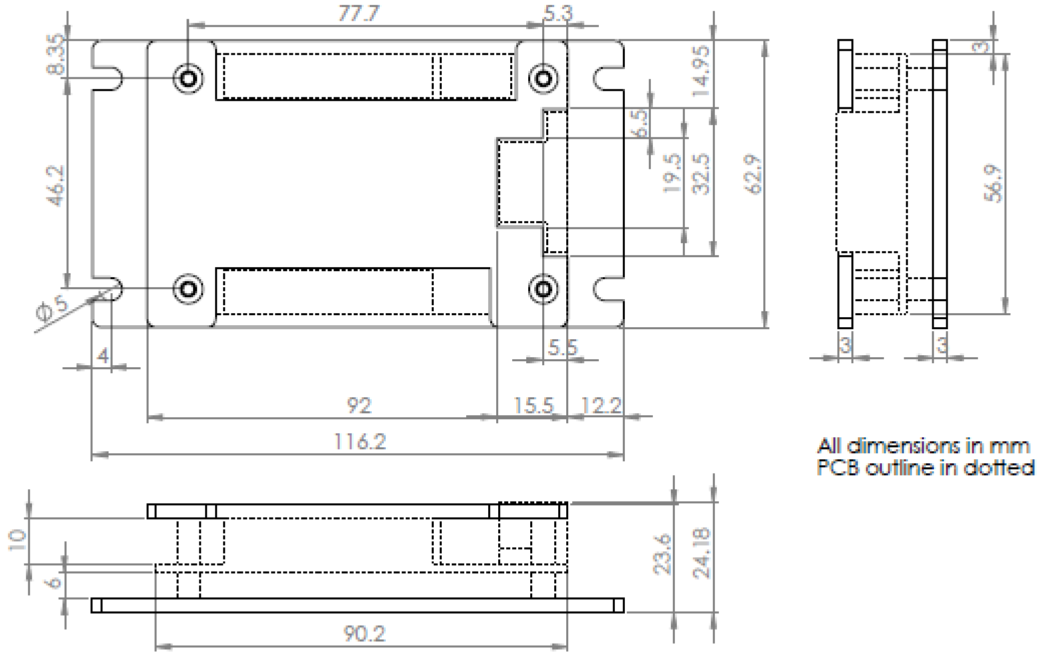

Figure 12: MK-200C package outline

4. Software Tools

A free software tool, the MOTKA Motion Companion, can be downloaded from our official website. This tool is designed to help user to learn and use all the supported motion controllers developed by MOTKA.

Please refer to the MOTKA Motion Companion for details.

5. Command Basics

A list of essential, straight-forward commands is developed to support fundamental motion control needs.

Please refer to the Command Reference for details.

6. Revision History

Date	Revision	Changes
09-April-2015	1.0	Initial release.
18-Feb-2016	1.1	Added Section 2.9 – Design Examples.

Table 2 - Document revision history