CONTENTS

BASICS

Definition

Mark Roberts Protocol (MRP) is the communication system available for control software that wishes to interact via Ethernet with the Mark Roberts Motor Control Programs that run on the various motor control boards manufactured by Mark Roberts.

It is a UDP protocol.

The Mark Roberts Motor Control Programs are described collectively in this document as MR_ROBOTICS.

The protocol was developed by Mark Roberts Motion Control Ltd. Mark Roberts offer various applications that use **MRP** to communicate with MR_ROBOTICS, but the protocol is also published for use by other application developers.

Start Up

Architecture 1

In this architecture, MR_ROBOTICS is pre-loaded into Flash memory and as soon as one powers up the axis board, the application starts running.

Architecture 2

When a motor board, also referred to as an axis board, is switched on, it automatically starts executing a program known as the boot loader. This program allows MR_ROBOTICS to be transmitted to the axis board via Ethernet, and the board then automatically starts executing MR_ROBOTICS.

NETWORK COMMUNICATION

Application software communicates with MR_ROBOTICS using UDP protocol. Four network sockets are used.

Network Sockets

Each network socket has a particular network port and a particular ip address. (There is a special case where one can set the ip address to ***broadcast*** which means that packets transmitted through such a socket are sent to every ip address on the network).

Network Ports

MR_ROBOTICS uses the following network ports:

25000
MR_ROBOTICS transmits communications on this port.

25001
MR_ROBOTICS receives communications that are broadcast to all axis boards on this port. MR_ROBOTICS will respond to a PING command that is received on this port, thus allowing application software to find out the ip address of all Mark Roberts axis boards on a given network.

25002
MR_ROBOTICS receives communications intended for one individual axis board only on this port. Most received communications are of this category.

25003
To send a reset to MR_ROBOTICS, you send a packet to this port. The contents of the packet are irrelevant. ANY communication received on port 25003 will generate a reset.

IP Addresses

When an axis board is released from the Mark Roberts factory, it is set up with IP address 192.168.1.236. This may be changed at any time, using the WRITE_NETWORK command.

Examples

On a system with 1 axis board, these are the sockets that an application communicating to MR_ROBOTICS must open:

One socket using port 25000. MR_ROBOTICS uses this socket to communicate back to the application

One socket using port 25001 to broadcast packets. MR_ROBOTICS expects to receive communications from the application on this socket.

One socket using port 25002 and the ip address of the axis board. MR_ROBOTICS expects to receive communications from the application on this socket.

Conditional:

If you want to be able to restart the application without having to switch the axis board off and on, then you also need one socket using port 25003 and the ip address of the axis board.

On a system with 2 axis boards, these are the sockets that would need to be open:

One socket using port 25000. Each board running MR_ROBOTICS uses this socket to communicate back to the application.

One socket using port 25001 to broadcast packets. MR_ROBOTICS expects to receive communications from the application on this socket.

Two sockets, each using port 25002, one using the ip address of the first axis board and the other using the ip address of the second axis board. MR_ROBOTICS expects to receive communications from the application on this socket.

Conditional:

If you want to be able to restart the application without having to switch the axis board off and on, then you also need one socket using port 25003 with the ip address set to ***broadcast***. Alternatively, you can open 2 sockets each using port 25003, one using the ip address of the first axis board and the other using the ip address of the second axis board.

MOTOR CONTROL

The main and fundamental task of MR_ROBOTICS is to control the movement of axes. Each board running MR_ROBOTICS can control several different axes. The maximum no. of axes is 4 for the Mark Roberts Quad board, 6 for the Mark Roberts Hex board, 8 for the Mark Roberts Octo board or 12 for the Mark Roberts Ulti board.

To accomplish this axis control, MR_ROBOTICS supports 3 modes of operation. These are known as *position* mode, *velocity* mode and *independent* mode.

Position Mode

The axis is moved through the use of the POSITION command

When an external application sends a POSITION command, MR_ROBOTICS will move the axis from its current position to the newly received position in 1/50 of a second. During this time, MR_ROBOTICS performs mathematics which monitors motor position and smoothness 32 times during this 1/50 of a second transition.

If no POSITION command is received then MR_ROBOTICS considers the desired position to remain the same. It is therefore not necessary to keep sending POSITION commands continuously if one wishes an axis to be stationary.

To accomplish movement over say 10 seconds, one would send 500 POSITION commands to MR_ROBOTICS at 1/50 second intervals.

Velocity Mode

The axis is moved through the use of the VELOCITY command.

When an external application sends a VELOCITY command, MR_ROBOTICS will add the distance that will be accomplished by that specified velocity to the current axis position and move the axis to this new position in 1/50 of a second. During this time, MR_ROBOTICS performs mathematics which

monitors motor position and smoothness 32 times during this 1/50 of a second transition.

If the requested distance cannot be accomplished in 1/50 second without exceeding the maximum allowed acceleration, then the existing velocity will be increased by the maximum acceleration each 1/50 second until the requested velocity is reached.

Velocities may be sent at any time. The velocity being applied is changed by sending a new Velocity command with a different value. Stopping is accomplished by sending a zero velocity value.

Independent Mode

The axis is moved through the use of the GOTO command.

The GOTO command has a destination position and a duration value. MR_ROBOTICS then automatically moves to the specified destination in the specified duration, generating positions itself 50 times per second for the duration of time specified.
During each 1/50 second interval, MR_ROBOTICS performs mathematics which monitors motor position and smoothness 32 times during this 1/50 of a second transition

The duration is specified in fiftieths (1/50) of a second. If a value of zero is put as the duration, then MR_ROBOTICS will compute a duration based on the maximum velocity, maximum acceleration and goto “speed factor”. The goto “speed factor” is a value saying at what percentage of maximum possible speed, a given goto should be executed.

A movement being executed in independent mode may be halted by sending a STOP command.

THE PROTOCOL

Classes of Commands & Responses

All commands and responses have a 12 byte header.
The following classes of command and response exist

Class 1 These use the header only.
Class 2 These contain data for multiple axes.
Class 3 These contain more data than contained in the header but do not contain data for multiple axes.

Packets Coming In To MR_ROBOTICS		
Class 1	**Class 2**	**Class 3**
STARTBOARD	POSITION	WRITE NETWORK
PING	VELOCITY	WRITE AXIS
HOME ZERO	PV	LOAD AXIS
STOP	GOTO	LOAD TUNINGS
DIRECT ZERO	TRACK	TEN PIN
ENABLE AXIS	WRITE SETTING	VISCA
DISABLE AXIS	LOAD SETTING	TRACK PID
OVERRIDE LIMITS		ZOOM TABLE
RESTORE LIMITS		
SET TRIGGER		
CLEAR TRIGGER		
READ NETWORK		
READ AXIS		
FACTORY		
ZOOM RELATED SCALING		
READ SETTING		
READ METRICS		
FLAIR		
ZOOM TABLE ENABLE		

Packets Going Out From MR_ROBOTICS		
Class 1	**Class 2**	**Class 3**
STARTBOARD	POSITION	PING
HOME ZERO	READ SETTING	READ AXIS
		READ NETWORK
		READ METRICS
		TEN PIN
		VISCA
		ZOOM TABLE ENABLED STATUS

Class 1 Commands & Responses

These use the header only.

```
typedef struct
{
      unsigned func  : 8;       // Function code
      unsigned board : 8;       // Board ID  max 32
      unsigned axis  : 8;       // Axis number
      unsigned subf  : 8;       // Sub-function code
      uint16   seq;             // packet sequence number
      uint16   size;            // Size of payload AND header
                                // (i.e. payload size + 8)
      uint16   code;            // general purpose
                                // function-dependent value
      uint16   reserved;
} pktheader;

#define PKT_HDR_SIZE     sizeof (pktheader)       // 12
```

The Class 1 Commands are:

STARTBOARD
PING
HOME ZERO
STOP
DIRECT ZERO
ENABLE AXIS
DISABLE AXIS
OVERRIDE LIMITS
RESTORE LIMITS
SET TRIGGER
CLEAR TRIGGER
READ NETWORK
READ AXIS
FACTORY
ZOOM RELATED SCALING
READ SETTING
READ METRICS
FLAIR
ZOOM TABLE ENABLE

Class 2 (Multiple Axis) Commands & Responses

These contain data for multiple axes.

After the 12 byte header, multiple axis commands have a 24 byte sub-header, followed by 4 bytes of data for each axis.

```
typedef struct
{
      int32[4] reserved;
      uint16 axisflags;
      uint16 spare;
      int32 duration;
      axdndata axdndats[MAX_AXES];
} multidndata;

#define MULTI_DN_HDR_SIZE (5 * sizeof(int32))      // 24

typedef struct
{
      union
      {
            int32  lMovecmd;     // e.g. position, velocity
            float  fMovecmd;     // e.g. position, velocity

      };
} axdndata;
```

The 7 multiple axis commands are:

POSITION
VELOCITY
PV
GOTO
TRACK
WRITE SETTING
LOAD SETTING

After the 12 byte header, multiple axis responses have a 24 byte sub-header, followed by 8 bytes of data for each axis.

```
typedef struct
{
      uint16        network_timer;
      uint16        axisflags;
      int32[5]      reserved;
      axupdata axupdats[MAX_AXES];
} multiupdata;

#define MULTI_UP_HDR_SIZE (5 * sizeof(int32))            // 20

typedef struct
{
      union
      {
            int32  lPosn;        // Signed motor position in slots
            float  fPosn;        // Motor position in scaled units
      };
      uint32 limitstates : 4;    // Limit-1 Limit-2 Limit-3 Tripped-Status
      uint32 whytripped  : 6;    // Why the axis tripped
      uint32 reserved1   : 6;
      uint32 reserved2   : 2;
      uint32 reserved3   : 6;
      uint32 spare       : 8;
} axupdata;
```

The 2 multiple axis responses are:

POSITION
READ SETTING

Class 3 Commands & Responses

These contain more data than contained in the header but
do not contain data for multiple axes.

The following format is used by the WRITE_NETWORK command, by the PING response and by the READ_NETWORK response

```
typedef struct _ipaddr
{
	unsigned addr0  : 8;
	unsigned addr1  : 8;
	unsigned addr2  : 8;
	unsigned addr3  : 8;
} ipaddr;

typedef struct
{
	ipaddr ip_address;
	uint16 reserved;
	uint16 reserved;
	ipaddr subnet_mask;
	ipaddr default_gateway;
	uint16 mac_address[6];
	uint16 MRMC board type;
		881 = Hex
		882 = Hex in Nikon Pod

	uint16 boardver;
	uint16 progcode
	uint16 progversion;
} pingdata;
```

The following format is used by the WRITE_AXIS command, the LOAD_AXIS command, the LOAD_TUNINGS command and the READ_AXIS response. (For the LOAD TUNINGS command, only the signal gain, integral gain & tacho gain fields are relevant).

```
typedef struct
{
	int32	reserved;
	unsigned	motortype		:8
	unsigned	homingstyle		:8
	unsigned	reserved		:8
	unsigned	reserved		:8
	unsigned	reserved		:8
	unsigned	reserved		:8
	unsigned	reserved		:8
	unsigned	reserved		:8
	float	gearing;
	float	maxacc;
	float	maxvel;
	float	minpos;
	float	maxpos;
	float	reserved;
	float	homing_velocity;
	float	homing_time;
	float	homing_offset;
	float	backlash_offset;
	int32	signal_gain;
	int32	integral_gain;
	int32	tacho_gain;
	int32	current_limit;
	int32	temperature_limit;
	int32	position_error_limit;
	int32	steplength;
	int32	pwmtype;
	float[18] reserved;
} motordata;
```

LIST OF COMMANDS AND RESPONSES

In addition to the main POSITION, VELOCITY, GOTO and STOP commands, there are a number of other supporting commands in MRP which provide additional functions, when these are needed.

Here are 2 lists.

Firstly a list of each of the MRP packets that can come in to MR_ROBOTICS.

Secondly a list of each of the MRP responses that can be received from MR_ROBOTICS.

PACKETS COMING IN TO MR_ROBOTICS

STARTBOARD

Code	Class	Name	Length	Description
01 0x01	1	STARTBOARD	12	The STARTBOARD packet should be sent on start-up. MR_ROBOTICS waits for a STARTBOARD packet before enabling any control of motors. When the STARTBOARD packet arrives, MR_ROBOTICS reads in the motor configuration data from the Flash memory of the board, starts motor control, and sends a STARTBOARD response packet with the return code HAVE STARTED. (refer "Packets Going Out From MR_ROBOTICS" section) If a STARTBOARD packet is sent when MR_ROBOTICS has already started motor control – then MR_ROBOTICS sends a STARTBOARD response packet with the return code ALREADY STARTED.

Position	Bytes	Format	Description
0	1	Character	Command Code = 1
1	1	Integer	Board number (optional according to application)
2	1		Zoom axis number This is required when the MRMC board is controlling a zoom axis and the customer is using either the zoom related scaling feature (ZRS) or the zoom linearization feature. The value is 1 based so if the zoom is axis 1 then the value should also be 1 and so on.
3	1	Integer	Network buffer size Used when in Position Mode

			Unit is 20 milliseconds e.g. 1 = 20 milliseconds e.g. 2 = 40 milliseconds e.g. 3 = 60 milliseconds Recommended value is 2
4-5	1	Bit flags[16]	System Flags Bit 0 Board Sync Used by position mode. Synchronises each board to the controller at an accuracy of 125 microseconds 0 = Board sync On 1 = Board sync off If intending to use position mode, set this to On. If intending to use only velocity and independent modes, set this to Off Bit 1 Nikon Auto-focus With Nikon DSLR cameras one must release the auto-focus after taking a picture before another picture may be taken. 0 = Not using Nikon camera 1 = Using Nikon camera Bit 2 Preset Storage Location Preset positions may be stored either on the controlling system or on the Mark Roberts robot 0 = Presets are stored on the controlling system 1 = Presets are store on the Mark Roberts robot Bit 3 Soft Limit Control Axis limits may be monitored either by the controlling software or by the Mark Roberts robot itself

			0 = Axis limits are monitored by the controlling software 1 = Axis limits are monitored by the Mark Roberts robot Bit 4 Relevant when using Atlas motors 0 = Atlas motors execute presets by using velocities generated by the controlling software 1 = Atlas motors execute presets by passing a single destination position to the drive. Bits 5-15 Reserved
6-7	2	Integer	Size (12)
8	1		Focus axis number This is required when the MRMC board is controlling a focus axis and the customer is using the zoom related scaling feature (ZRS). ZRS must not be applied to the focus axis The value is 1 based so if the focus is axis 1 then the value should also be 1 and so on.
9	1		Iris axis number This is required when the MRMC board is controlling an iris axis and the customer is using the zoom related scaling feature (ZRS). ZRS must not be applied to the iris axis The value is 1 based so if the iris is axis 1 then the value should also be 1 and so on.
10-11	2		Reserved

PING

Code	Class	Name	Length	Description
2 0x02	1	PING	12	Send a communication to see whether there is a Mark Roberts board at a given ip address. When MR_ROBOTICS receives a PING command, it returns a PING packet giving the ip address of the board on which it is running, together with subnet mask, default gateway and version information - for format see PING response under ("PACKETS GOING OUT FROM MR_ROBOTICS") In order for MR_ROBOTICS to return a PING response, network sockets must be open as outlined below & in the **Network Sockets** section of this manual. The application talking to MR_ROBOTICS should have a socket open using port 25002 for each axis board. E.g. if you had 2 axis boards set to ip addresses 192.168.1.236 and 192.168.1.237 then you would have 2 sockets open One: port 25002 ip addr 192.168.1.236 Two: port 25002 ip addr 192.168.1.237 You transmit a PING command on each of these sockets. If MR_ROBOTICS has not yet successfully received any communication via port 25002, then it will respond to a PING command that is sent via a socket opened for broadcast using port 25001. No other commands are effective via this mechanism. This allows an application to find the IP address of a board, should this be unknown. The application should also have a socket open using port 25000. Each board running MR_ROBOTICS will send its PING response on this socket.

Position	Bytes	Format	Description
0	1	Character	Command Code = 2

1	1	Integer	Board number (optional according to application)
2	1		Reserved
3	1		0 = BASIC PING RESPONSE - does not return a name 1 = EXTENDED PING RESPONSE - returns a name
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

HOME ZERO

Code	Class	Name	Length	Description
10 Ox0A	1	HOME	12	Assigns a specified position to an axis by a variety of methods. Establishes a fixed reference point for an axis via sensors or other means of detecting end of travel and then assigns that axis a position. The position to be assigned is held in Flash memory (see AXIS SAVE). It is most commonly zero but doesn't have to be.

Position	Bytes	Format	Description
0	1	Character	Command Code = 10
1	1	Integer	Board number (optional according to application)
2	1	Integer	Axis number to be homed – range 0-16, where 1-16 means home a single axis and 0 means home multiple axes, using the mask in bytes 8-9 to determine which axes
3	1	Integer	Relevant when homing multiple axes 0 = serial - home each selected axis in turn 1 = parallel - home all selected axes simultaneously
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-9	2		Flags indicating whether each axis is to be homed (only relevant if byte 2 is 0) One bit for each of the 16 possible axes on a board. Axis 1 uses the lowest bit; axis 16 uses the highest
10-11	2		Reserved

POSITION

Code	Class	Name	Length	Description
11 0x0B	2	POSITION	36 + (MAX_AXES * 4)	Sends desired positions for each of up to 16 axes on a board.

Position	Bytes	Format	Description
0	1	Character	Command Code = 11
1	1	Integer	Board number (optional according to application)
2-3	2		Reserved
4-5	2	Integer	Sequence Number Each POSITION command that is sent is given a consecutive sequence number. MR_ROBOTICS uses these to detect dropped packets and is then able to interpolate positions where such are detected.
6-7	2	Integer	Size 12 + 24 + (4 * AXES_PER_BOARD) e.g. 52 for 4 axis board e.g. 68 for 8 axis board
8-11	4		Reserved
12-35	24		Reserved
36 - 39	4	Floating Point	Desired position for first axis
40 - end	4	Floating Point	Desired position for each subsequent axis

VELOCITY

Code	Class	Name	Length	Description
12 0x0C	2	VELOCITY	36 + (MAX_AXES * 4)	Sends desired velocities for each of up to 16 axes on a board.

Position	Bytes	Format	Description
0	1	Character	Command Code = 12
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 12 + 24 + (4 * AXES_PER_BOARD) e.g. 52 for 4 axis board e.g. 68 for 8 axis board
8-11	4		Reserved
12-35	24		Reserved
36 - 39	4	Floating Point	Desired velocity for first axis
40 - end		Floating Point	Desired velocity for each subsequent axis

PV

Code	Class	Name	Length	Description
13 0x0D	2	PV	36 + (MAX_AXES * 4)	Sends desired velocities or positions for each of up to 16 motors on a board. This command is used in systems with lens axes which may be controlled by either velocity or by position, according to circumstance.

Position	Bytes	Format	Description
0	1	Character	Command Code = 13
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 12 + 24 + (4 * AXES_PER_BOARD) e.g. 52 for 4 axis board e.g. 68 for 8 axis board
8-9	2		Axis mask A value of 0 = velocity A value of 1 = position Example: Value of 0x04 if Axis 3 is controlled by position and all other axes are controlled by Velocity Example: Value of 0x06 if axes 2 & 3 are controlled by position and all other axes by velocity Example: Value of 0x00 if all axes are controlled by Velocity Note that use of positional control via the PV command is only permitted for lens axes

10-11	16		Reserved
12-35	24		Reserved
36 - 39	4	Floating Point	Desired velocity or position for first axis
40 - end		Floating Point	Desired velocity or position for each subsequent axis

GOTO

Code	Class	Name	Length	Description
14 0x0E	2	GOTO	36 + (MAX_AXES * 4)	Move the axes from their current positions to the destination positions specified. This command is only acted on if all axes are stationary

Position	Bytes	Format	Description
0	1	Character	Command Code = 14
1	1	Integer	Board number (optional according to application)
2	1		Axis number Relevant when using a Nikon camera. Zero based – range 0-15 If the focus characteristic in byte 3 is set to 1, then this axis will be home zeroed prior to execution of the goto, provided its homing type is set to 9 (Slipping Clutch style).
3	1		Focus Characteristic Relevant when using a Nikon camera. If using any camera other than Nikon then focus characteristic should = 0 If using auto focus on a Nikon camera then focus characteristic should = 0 If using a Nikon camera with a fixed focus point then focus characteristic should = 1
4-5	2		Reserved

6-7	2	Size	Size 12 + 24 + (4 * AXES_PER_BOARD) e.g. 52 for 4 axis board e.g. 68 for 8 axis board
8	1		If byte 9 contains 0xFF then the value here specifies the number of times that the GOTO will repeat. E.g. a value of 1 causes the robot to move to the destination position & then return to the start position, once. A value of -1 means that GOTOs will continue to repeat until STOP is pressed. A value of 0 is invalid and no GOTO will be executed
9	1		If this byte is 0xFF then byte 8 contains the number of times that this GOTO should repeat. Any other value than 0xFF results in a standard GOTO which simply goes to the specified destination position.
10-11	2		Reserved
12-27	16		Reserved
28-31	4		Goto speed factor expressed as a fraction of maximum allowed speed. e.g. 100% = 1.0 50% = 0.5 10% = 0.1
32 - 35	4	Integer	Duration in which the GOTO is to be executed, specified in fiftieths of a second. E.g. 250 = 5 seconds. If a value of zero is specified, MR_ROBOTICS will compute a duration based on the maximum velocity, maximum acceleration and goto “speed factor” of the axes. (see WRITE_AXIS command).
36 - 39	4	Floating	Destination position for first axis

		Point	
40 - end		Floating Point	Destination position for each subsequent axis

TRACK

Code	Class	Name	Length	Description
16 0x10	2	TRACK	36 + (MAX_AXES * 4)	Sends a tracking parameter for each of up to 16 axes on a board.

Position	Bytes	Format	Description
0	1	Character	Command Code = 16
1	1	Integer	Board number (optional according to application)
2	1		Reserved
3	1		Type of tracking parameter 0 = tracking error 1= tracking target position
4-5	2		Reserved
6-7	2	Integer	Size 12 + 24 + (4 * AXES_PER_BOARD) e.g. 52 for 4 axis board e.g. 68 for 8 axis board
8-11	4		Reserved
12-35	24		Reserved
36 - 39	4	Floating Point	Tracking parameter for first axis
40 - end		Floating Point	Tracking parameters for each subsequent axis

MOVE

Code	Class	Name	Length	Description
17 0x11	2	MOVE	12 + 32 + ((MAX_AXES * 4)) *8)	Sends move data for up to 8 positions for each of up to 16 axes on a board.

Position	Bytes	Format	Description
0	1	Character	Command Code = 17
1	1	Integer	Board number (optional according to application)
2-3	2		Reserved
4-5	2		The total number of positions specified for the move (range 3-8)
6-7	2	Integer	Size 44 + ((4 * AXES_PER_BOARD) * 8) e.g. 172 for 4 axis board e.g. 300 for 8 axis board
8-11	4		Reserved
12 - 43	28	Floating Point	Absolute time (1/50 second) for each position e.g. 10 second move with 3 points at 0,5 and 10 seconds, would send time values of 0,250 and 500.
44 - 47	4	Floating Pt	Start position for first axis
48 - 75	28	Floating Point	Subsequent positions for first axis If a move has 3 points then the last 5 elements (bytes 80-99) are irrelevant If a move has 4 points then the last 4 elements are irrelevant, and so on.
76-end		Floating Point	Desired start position and subsequent positions for each subsequent axis.

STOP

Code	Class	Name	Length	Description
15 0x0F	1	STOP	12	Stop a GOTO that is in progress

Position	Bytes	Format	Description
0	1	Character	Command Code = 15
1	1	Integer	Board number (optional according to application)
2-3	2		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

DIRECT ZERO

Code	Class	Name	Length	Description
20 0x14	1	ZERO	12	Directly assigns a specified position to an axis. The command is known as ZERO because the position assigned is most commonly zero. The position to be assigned is held in Flash memory (see AXIS SAVE).It is usually zero but doesn't have to be.

Position	Bytes	Format	Description
0	1	Character	Command Code = 20
1	1	Integer	Board number (optional according to application)
2	1	Integer	Axis no. whose current position is to be set. With DC motors, this places a value in the D2A. With stepper motors it is simply a software reference point. Axis number to be assigned a position – range 0-16, where 1-16 means assign a position to a single axis and 0 means assign a position to multiple axes, using the mask in byte 3 to determine which axes
3	1		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

ENABLE AXIS

Code	Class	Name	Length	Description
21 0x15	1	ENABLE	12	Enable the specified axis. A axis may be turned off and on by the application. Where the motor type is such that MR_ROBOTICS can determine error conditions, then these will override the enable command.

Position	Bytes	Format	Description
0	1	Character	Command Code = 21
1	1	Integer	Board number (optional according to application)
2	1	Integer	Axis number to be enabled – range 0-15
3	1		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

DISABLE AXIS

Code	Class	Name	Length	Description
22 0x16	1	DISABLE	12	Disables the specified axis An axis may be turned off and on by the application. (Where the motor type is such that MR_ROBOTICS can determine error conditions, then axes can also be automatically disabled by MR_ROBOTICS, as a result)

Position	Bytes	Format	Description
0	1	Character	Command Code = 22
1	1	Integer	Board number (optional according to application)
2	1	Integer	Axis number to be disabled – range 0-15
3	1		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

OVERRIDE LIMITS

Code	Class	Name	Length	Description
23 0x17	1	OVERRIDE LIMITS	12	Limit switches can be fitted so that a motor cannot drive equipment beyond a safe point – e.g. a limit switch could be fitted where a motor was driving a camera dolly along some track, so that the dolly could not be driven beyond the end of the track. In some circumstances, one can temporarily disable such limit switches. E.g. if you want to use reaching the limit switch as a mechanism for establishing a consistent home position. Each time you power up the equipment, the dolly can be instructed to advance to a limit switch and then back off by a fixed number of turns of the motor. When executing this procedure, you do not want the motor to cut out when you reach the limit switch, so you send an OVERRIDE LIMITS command

Position	Bytes	Format	Description
0	1	Character	Command Code = 23
1	1	Integer	Board number (optional according to application)
2	1	Integer	Number of axis whose limits are to be overridden – zero based - range 0-15
3	1		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

RESTORE LIMITS

Code	Class	Name	Length	Description
24 0x18	1	RESTORE LIMITS	12	Used after one has issued an OVERRIDE LIMITS command to restore limit switches to normal operation

Position	Bytes	Format	Description
0	1	Character	Command Code = 24
1	1	Integer	Board number (optional according to application)
2	1	Integer	Number of axis whose limits are to be restored – zero based - range 0-15
3	1		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

SET TRIGGER

Code	Class	Name	Length	Description
31 0x1F	1	SET TRIGGER	12	Turns on a trigger output

Position	Bytes	Format	Description
0	1	Character	Command Code = 31
1	1	Integer	Board number (optional according to application)
2	1	Integer	Trigger number – range 0-2
3	1		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

CLEAR TRIGGER

Code	Class	Name	Length	Description
32 0x20	1	CLEAR TRIGGER	12	Turns off a trigger output

Position	Bytes	Format	Description
0	1	Character	Command Code = 32
1	1	Integer	Board number (optional according to application)
2	1	Integer	Trigger number – range 0-2
3	1		Reserved
4-5	2		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

WRITE NETWORK

Code	Class	Name	Length	Description
41 0x29	3	BOARD SAVE	56	Permanently change the IP address that the board is using MR_ROBOTICS closes the existing socket, opens a new one with the specified ip address & then saves this new ip address in flash memory so it is remembered even after the board has been powered down. Note that this command also saves the subnet mask and default gateway values to Flash:

Position	Bytes	Format	Description
0	1	Character	Command Code = 41
1	1	Integer	Board number (optional according to application)
2			Reserved
3		Character	0xCD
4-5		Integer	0x89AB
6-7	2	Integer	Size (56)
8-11	4		Reserved
12-15	4	Ip address	1st byte of ip address e.g. 192 2nd byte of ip address e.g. 168 3rd byte of ip address e.g. 1 4th byte of ip address e.g. 236
18-19	2		Reserved
20-23	4	Ip address	1st byte of subnet mask e.g. 255 2nd byte of subnet mask e.g. 255 3rd byte of subnet mask e.g. 255 4th byte of subnet mask e.g. 0

24-27	4	Ip address	1st byte of default gateway e.g. 192 2nd byte of default gateway e.g. 168 3rd byte of default gateway e.g. 1 4th byte of default gateway e.g. 1
28-39	12	MAC address	MAC address (not currently used)
40-55	16		Reserved

READ NETWORK

Code	Class	Name	Length	Description
42 0x2A	1	READ NETWORK	12	Asks for the network data stored in flash memory to be sent back in a READ NETWORK response packet

Position	Bytes	Format	Description
0	1	Character	Command Code = 42
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 12
8-11	4		Reserved

WRITE AXIS, LOAD AXIS and LOAD TUNINGS

There are 2 architectures for storing axis configuration data.

1. The board does not retain axis configuration data. Instead the controlling software loads this data to the board each time that the board is powered up.

2. The board retains axis configuration data in its Flash memory.

Code	Class	Name	Length	Description
43 0x2B	3	WRITE AXIS	168	Configuration data for the specified axis. This is written into FLASH memory and remains in place permanently. Note that the precise sequence of actions is: 1. Configuration data for all axes is read from flash memory into the axis data area in RAM 2. For the specified axis, the data within the WRITE AXIS packet overwrites the axis data area in RAM 3. All of the axis data area is written to FLASH memory

Code	Class	Name	Length	Description
47 0x2F	3	LOAD AXIS	168	Configuration data for the specified axis. This is loaded into RAM and remains in place until the BOARD is powered down

Code	Class	Name	Length	Description
49 0x31	3	LOAD TUNINGS	168	This is the same as LOAD AXIS but only the header data (bytes 0-11) and the tuning data (bytes 60-71) are used. All other fields are ignored.

Position	Bytes	Format	Description
0	1	Character	Command Code = 43 (WRITE AXIS) Command Code = 47 (LOAD AXIS)
1	1	Integer	Board number (optional according to application)

2	1	Integer	Axis number to be configured – range 0-15
3-5			Reserved
6-7	2	Integer	Size (168)
8-9	2	Integer	For a WRITE AXIS command to be obeyed, these 2 bytes must contain 0x579A
10-11	2		Reserved
12-15	4		Reserved
16	1	Character	Motor Type 0 = Servo (External) 1 = Stepper 2 = DtoA 3 = Serially Controlled Iris 4 = Serially Controlled Focus 5 = Serially Controlled Zoom 6 = Servo (PWM) 7 = CAN
17	1	Character	Homing Style 1 = Search for limit 1 sensor (zero marker pulse available) 2 = Search for limit 2 sensor (zero marker pulse available) 3 = Search for limit 2 sensor (bi-directional) This method can be used with axes which rotate such as pan, tilt or turntable. It optimises homing time by optically detecting in which half of the travel the axis resides, and setting the sign of the homing velocity to be that which will reach the sensor soonest. 4 = Search for limit 1 sensor (no zero marker pulse available) 5 = Search for limit 2 sensor (no zero marker pulse available) 6 = Search for limit 2 sensor

			(no zero marker pulse available) (bi-directional – see style 3 above) 7 = Lens Calibration Detects each end of travel of a lens 8 = As per 6, but, if on the sensor, does not check for the sensor for the first 10 seconds of searching. Required for standalone turntables. 9 = Slip Zero Moves at the homing velocity until the homing time has passed after which the axis moves to the position specified in the homing offset 10 = Direct The current position of the axis is set as zero 11 = Absolute Used when an axis is fitted with an absolute encoder and is thus able to report position without need of homing.
18	1	Character	Spare
19	1	Character	Goto Style 0 = All axes reach destination at same time 1 = NOT YET IMPLEMENTED (Each axis reaches its destination as swiftly as it can)
20	1	Character	Automatically Home on Start-up? 0 = No 1 = NOT YET IMPLEMENTED (Yes)
21-23	3		Reserved
24-27	4	Floating Point	Gearing ratio. This allows the controlling software to send position and velocity values in meaningful units such as degrees or inches etc.,
28-31	4	Floating Point	Maximum Acceleration

32-35	4	Floating Point	Maximum Velocity
36-39	4	Floating Point	Minimum Position NOT YET IMPLEMENTED
40-43	4	Floating Point	Maximum Position NOT YET IMPLEMENTED
44-47	4	Floating Point	Maximum Change of Acceleration Factor The maximum permitted change of acceleration expressed as a fraction of the maximum acceleration. E.g. if max vel = 200, max acc = 50 & max change of acc factor = 0.2 then the velocities and accelerations of succeeding ticks will be: Vel Acc 0 10 10 20 30 30 60 40 100 50 150 50 200 0 200 0 On light equipment the maximum change of acceleration factor may be 1.0 but on heavier equipment greater smoothness is achieved if it is lowered.
48-51	4	Floating Point	Homing Velocity The velocity at which the axis seeks for its reference sensor
52-55	4	Floating Point	Homing Time Number of seconds allowed for the axis to seek for its reference sensor
56-59	4	Floating Point	Homing Offset

			After homing has calibrated the axis from the sensor, the axis can then be moved from that calibration position to a more convenient home position, offset by this value from the calibration position.
60-63	4		Backlash Offset This value is used where an axis has backlash. When an axis is moving in one direction, the specified offset is applied to compensate for the backlash. In the other direction, the backlash offset is not applied.
64-67	4	Integer	Signal gain (motor types SERVO or PWMSERVO)
68-71	4	Integer	Tacho gain (motor types SERVO or PWMSERVO)
72-75	4	Integer	Integral gain (motor types SERVO or PWMSERVO)
76-79	4	Integer	Current limit (motor types CAN, SERVO or PWMSERVO)
80-83	4	Integer	Temperature limit (motor type SERVO)
84-87	4	Integer	Positional Error Limit (all motor types)
88-91	4	Integer	Stepper pulse length (motor type STEPPER)
92-95	4	Integer	PWM type (motor types SERVO or PWMSERVO)
96-99	4	Floating Point	Maximum Deceleration
100-103	4	Floating Point	Maximum Change of Deceleration Factor
104-167	32		Reserved

READ AXIS

Code	Class	Name	Length	Description
44 0x2C	1	READ AXIS	12	Asks for axis setup data. Depending on the contents of byte 3, this may be asking for the axis setup data stored in flash memory (SAVED DATA) or the axis setup data currently stored in RAM (IN USE DATA). The axis setup data is sent back in a READ AXIS response packet

Position	Bytes	Format	Description
0	1	Character	Command Code = 44
1	1	Integer	Board number (optional according to application)
2	1	Integer	Axis number to be read – range 0-15
3	3		MEMORY TYPE 0 = FLASH - Return the axis setup data stored in flash memory (SAVED AXIS DATA) 3 = RAM - Return the axis setup data currently stored in RAM (IN USE AXIS DATA).
6-7	2	Integer	Size (12)
8-11	4		Reserved

WRITE SETTING and LOAD SETTING

There are 2 architectures for storing the various settings that comprise axis configuration data.

1. The board does not retain axis configuration data. Instead the controlling software loads this data to the board each time that the board is powered up.

2. The board retains axis configuration data in its Flash memory.

Code	Class	Name	Length	Description
45 0x2C	2	WRITE SETTING	36 + (MAX_AXES * 4)	Configuration data for the specified setting. This is written into FLASH memory and remains in place permanently. Note that the precise sequence of actions is: 1. Configuration data for all settings is read from flash memory into the axis data area in RAM 2. For each axis, the data within the WRITE SETTING packet overwrites the data for that setting in RAM 3. All of the axis data area is written to FLASH memory

Code	Class	Name	Length	Description
48 0x2A	2	LOAD SETTING	36 + (MAX_AXES * 4)	Configuration data of a specified setting for all axes. E.G. the maximum acceleration values for each axis or specified axis. This is loaded into RAM and remains in place until the BOARD is powered down

Position	Bytes	Format	Description
0	1	Character	Command Code = 45 (WRITE SETTING Command Code = 48 (LOAD SETTING)
1	1	Integer	Board number (optional according to application)
2	1		Reserved

3	1	Format	Code for the Setting being saved 1 = MOTOR TYPE 4 = HOMING STYLE 11 = GEARING RATIO 12 = MAXIMUM ACCELERATION 13 = MAXIMUM VELOCITY 14 = MAXIMUM POSITION 15 = MINIMUM POSITION 16 = CHANGE OF ACCELERATION FACTOR 17 = HOMING VELOCITY 18 = HOMING TIME 19 = HOMING OFFSET 20 = BACKLASH OFFSET 21 = PWM TYPE 22 = MAXIMUM DECELERATION 23 = CHANGE OF DECELERATION FACTOR 31 = CURRENT LIMIT 32 = TEMPERATURE LIMIT 33 = POSITIONAL ERROR LIMIT
4-5	2		Reserved
6-7	2	Integer	Size 12 + 24 + (4 * AXES_PER_BOARD) e.g. 52 for 4 axis board e.g. 68 for 8 axis board
8-9	2	Integer	For a WRITE SETTING command to be obeyed, these 2 bytes must contain 0x579B
10-11	2		Reserved
12-35	24		Reserved
36 - 39	4	Where value in bytes 2-3 is between 11 and 30 Floating Point Otherwise Integer	Setting value for first axis e.g. if Format is 1 then this is the motor type for the first axis e.g. if Format is 2 then this is the maximum acceleration for the first axis
40 - end		Same as for bytes 36-39 above	Setting value for each subsequent axis

READ SETTING

Code	Class	Name	Length	Description
46 0x2E	1	READ SETTING	12	Asks for the axis data stored in flash memory to be sent back in a READ SETTINGS response packet

Position	Bytes	Format	Description
0	1	Character	Command Code = 46
1	1	Integer	Board number (optional according to application)
2	1	Character	MEMORY TYPE When an MRMC ROBOTICS board is delivered from the factory it is preloaded with factory default axis settings. These settings are stored in flash memory and loaded into RAM when the board is powered up. Controlling software can alter the flash memory settings using WRITE_SETTING or the current RAM settings using LOAD_SETTING. Either of these may be read back 0 = Read Flash Settings Returns the axis setting data stored in flash memory 3 = Read Ram Settings Returns the axis setting data stored in RAM
3	1	Character	Code for the Setting being read 1 = MOTOR TYPE 4 = HOMING STYLE 11 = GEARING RATIO 12 = MAXIMUM ACCELERATION 13 = MAXIMUM VELOCITY 14 = MAXIMUM POSITION 15 = MINIMUM POSITION 16 = GOTO SPEED FACTOR

			17 = HOMING VELOCITY 18 = HOMING TIME 19 = HOMING OFFSET 20 = BACKLASH OFFSET 21 = PWM TYPE 22 = MAXIMUM DECELERATION 23 = CHANGE OF DECELERATION FACTOR 31 = CURRENT LIMIT 32 = TEMPERATURE LIMIT 33 = POSITIONAL ERROR LIMIT
6-7	2	Integer	Size (12)
8-11	4		Reserved

READ METRICS

Code	Class	Name	Length	Description
51 0x33	1	READ METRICS	12	Asks for measurement values (metrics) from a head which includes sensors for such values.

Position	Bytes	Format	Description
0	1	Character	Command Code = 51
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size (12)
8-11	4		Reserved

FLAIR

Code	Class	Name	Length	Description
60 0x3C	1	FLAIR	12	This is a specialised packet sent by Flair, a long standing special effects control program provided by Mark Roberts Motion Control. On receiving this packet, MR_ROBOTICS goes into a mode where it can receive the software that Flair requires in order to run.

Position	Bytes	Format	Description
0	1	Character	Command Code = 60
1	1	Integer	Board number (optional according to application)
2	1		Reserved
3	1		Security Code 0xA5
4-5	2		0x89AB Security Code
6-7	2	Integer	Size 12
8-11	4		Reserved

TEN PIN

Code	Class	Name	Length	Description
63 0x3F	3	TENPIN	24	This is a specialised packet sent by controllers that are authorised to control the 10-pin camera interface of a Nikon DSLR

Position	Bytes	Format	Description
0	1	Character	Command Code = 63
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 24
8-11	4		Reserved
12-23	12		Payload containing camera command

VISCA

Code	Class	Name	Length	Description
67 0x43	3	VISCA	24	This is a specialised packet sent by controllers that are controlling cameras via the VISCA camera control protocol

Position	Bytes	Format	Description
0	1	Character	Command Code = 67
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 24
8-11	4		Reserved
12-23	12		Payload containing camera command

TRACK PID

Code	Class	Name	Length	Description
50 0x32	3	TRACK PID	32	

Position	Bytes	Format	Description
0	1	Character	Command Code = 50
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 32
8-11	4		Reserved
12-15	4	Floating Point (range 0-255)	Proportional term for PID loop used to optimise tracking input data
16-19	4	Floating Point (range 0-255)	Integral term for PID loop used to optimise tracking input data
20-23	4	Floating Point (range 0-255)	Differential term for PID loop used to optimise tracking input data
24-27	4	Floating Point (range 0-255)	Deadband – minimum positional error needed for tracking to execute
28-31	4	Reserved	

ZOOM TABLE

Code	Class	Name	Length	Description
64 0x40	3	ZOOM TABLE	228	

Position	Bytes	Format	Description
0	1	Character	Command Code = 64
1	1	Integer	Board number (optional according to application)
2	1	Byte	Lens Number
3	1	Byte	Number of segments in table Minimum 2. Maximum 9.
4-5	2		Reserved
6-7	2	Integer	Size 228
8-11	4		Reserved
12-15	4	Float	Start focal length of first segment in the zoom linearization table
16-19	4	Float	End focal length of first segment in the zoom linearization table
20-35	4	Float	The four co-efficient values for the cubic equation of the first segment
36-59	4	Float	Start focal length, end focal length & co-efficient values for the 2nd segment
60-227	4	Float	Start focal length, end focal length and co-efficient values for up to 7 more segments. Unused segments should be filled with binary zeroes

ZOOM TABLE ENABLE

Code	Class	Name	Length	Description
65 0x41	1	ZOOM TABLE ENABLE	12	Zoom Table Enable This is to switch ON/OFF zoom linearisation during GOTO

Position	Bytes	Format	Description
0	1	Character	Command Code = 65
1	1	Integer	Board number (optional according to application)
2	1		Reserved
3	1	Integer	ON / OFF flag. 0 = Run Zoom axis normally during GOTOs 1 = Linearise Zoom Axis during GOTOs, if a zoom table exists
4-5	2		Reserved
6-7	2	Integer	Size 12
8-11	4	Integer	Reserved

FACTORY

Code	Class	Name	Length	Description
40 0x28	1	FACTORY	12	Restore axis settings to their factory defaults

Position	Bytes	Format	Description
0	1	Character	Command Code = 40
1	1	Integer	Board number (optional according to application)
2	1		Axis number whose factory default settings are to be restored – range 0-16, where 1-16 means home a single axis and 0 means all axes
3-5	2		Reserved
6-7	2	Integer	Size 12
8-9	2	Integer	For a FACTORY command to be obeyed, these 2 bytes must contain 0x579A
10-11	2		Reserved

ZRS (Zoom Related Scaling)

Code	Class	Name	Length	Description
66 0x42	1	ZRS	12	Zoom Related Scaling This is a percentage applied to the incoming velocity, maximum acceleration and maximum change of acceleration for the pan and tilt axes, according to the position of the zoom.

Position	Bytes	Format	Description
0	1	Character	Command Code = 51
1	1	Integer	Board number (optional according to application)
2	1		Curve profile for the scaling. 0 = linear scaling 1 = squared scaling
3	1		Flag stating whether to apply different percentage scaling to velocity, maximum acceleration and maximum change of acceleration, or whether to take the velocity percentage and make the maximum acceleration percentage and the maximum change of acceleration percentage the same as this 0 = use different percentages 1 = use same percentage
4-5	2	Integer	ZRS Percentage. Examples: 100 = No scaling 50 = Divide incoming velocity by 2 when at the tightest zoom position 25 = Divide incoming velocity by 4 when at the tightest zoom position 10 = Divide incoming velocity by 10 when at the tightest zoom position

			1 = Divide incoming velocity by 100 when at the tightest zoom position 0 = No scaling If byte 3 is non zero then this percentage is also used to scale the maximum permitted acceleration and the maximum permitted change of acceleration
6-7	2	Integer	Size 12
8-9	2	Integer	ZRS Max Acceleration Percentage. Relevant if byte 3 is 0. Examples: 100 = No scaling 50 = Divide maximum acceleration by 2 when at the tightest zoom position 25 = Divide maximum acceleration by 4 when at the tightest zoom position 0 = No scaling
10-11	2		ZRS Max Change of Acc Percentage. Relevant if byte 3 is 0. Examples: 100 = No scaling 50 = Divide max change of acceleration by 2 when at the tightest zoom position 25 = Divide max change of acceleration by 4 when at the tightest zoom position 0 = No scaling

PACKETS GOING OUT FROM MR_ROBOTICS

STARTBOARD

Code	Class	Name	Length	Description
1 0x01	1	STARTBOARD	12	When a board starts up, it awaits a STARTBOARD command. When this is received, the STARTBOARD response packet is sent back to the controller The board settings are read in from the Flash memory of the board and a startboard response code of HAVE STARTED is returned. Bytes 8-9 contain a bit mask communicating which axes have saved settings stored in Flash, rather than the factory default values supplied with the system initially. If a STARTBOARD command is received when the firmware has already started then a startboard response code of ALREADY STARTED is returned and no action is taken by the firmware.

Position	Bytes	Format	Description
0	1	Character	Packet Code = 1
1	1	Integer	Board number (optional according to application)
2	1	Integer	Startboard Response Code 2 = HAVE STARTED 3 = ALREADY STARTED 12 = HAVE STARTED POD 13 = ALREADY STARTED POD
3	1		Board Version
4-5	2	Integer	Program version in 100 * format e.g. 107 = program version 1.7 e.g. 540 = program version 5.40
6-7	2	Integer	Size (12)

8-9	2	Integer	Bit mask for axes 1 = axis has setup data stored in flash memory on the axis board 0 = axis does not have setup data stored in flash memory on the axis board e.g. a value of 0x37, would indicate that axes 1-3 and axes 5-6 had axis setup data stored in flash memory
10-11	2	Integer	Bit mask for firmware type Bit 0 Encoder function 0 = encoder controlled by a motor 1 = encoder controlled by pan bars

PING

Code	Class	Name	Length	Description
2 0x02	3	PING	48 (BASIC PING RESPONSE) Or 78 (EXTENDED PING RESPONSE	This response is sent each time that MR_ROBOTICS receives a PING command

Position	Bytes	Format	Description
0	1	Character	Command Code = 2
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 48 (BASIC PING RESPONSE) or Size 78 (EXTENDED PING RESPONSE)
8-11	4		Reserved
12-15	8	Ip address	1st byte of ip address e.g. 192 2nd byte of ip address e.g. 168 3rd byte of ip address e.g. 1 4th byte of ip address e.g. 236
16-17	2	Integer	Maximum no. of axes that this board can control
18-19	2		Reserved
20-23	8	Ip address	1st byte of subnet mask e.g. 255 2nd byte of subnet mask e.g. 255 3rd byte of subnet mask e.g. 255 4th byte of subnet mask e.g. 0

24-27	8	Ip address	1st byte of default gateway e.g. 192 2nd byte of default gateway e.g. 168 3rd byte of default gateway e.g. 1 4th byte of default gateway e.g. 1
28-39	16		Reserved
40-41	2	Integer	Board Type 881 Hex Board 882 Pod
42-43	2	Integer	Board Version
44-45	2	Integer	Firmware code 210 Boot Loader 211 MR_ROBOTICS on Hex Board 212 MR_ROBOTICS on Quad Board 213 MR_ROBOTICS on Ulti Board 214 MR_ROBOTICS on Octo Board
46-47	2	Integer	Program version in 100 * format e.g. 107 = program version 1.7 e.g. 540 = program version 5.40
48-77	30	Name	An assigned name

HOME ZERO

Code	Class	Name	Length	Description
10 Ox0A	1	HOME ZERO	12	This packet is sent during homing either when a new stage is reached in the homing process, or when there is an error during homing, or when homing has successfully completed.

Position	Bytes	Format	Description
0	1	Character	Packet Code = 10
1	1	Integer	Board number (optional according to application)
2	1		Axis No. to which this homing message relates – zero based – range 0-15
3-5	3		Reserved
6-7	2	Integer	Size (12)
8-9	2	Integer	STATUS MESSAGES 0 = Completed homing successfully 1 = Invalid Axis Number 2 = Homing style is invalid 3 = Homing velocity is zero 4 = Time for finding sensor is zero 5 = Homing prohibited because the axis is moving 6 = Homing prohibited because already on sensor 7 = Invalid homing state 8 = Homing stopped by user 9 = Did not reach sensor in time 10 = Did not come off sensor in time 11 = Seeking Sensor 12 = Calibrating 13 = Moving to Offset 14 = Slipping a clutch

			21 = Could not find lens stop 1 22 = Could not find lens stop 2 31 = Could not find coarse level sensor 32 = Could not find fine level sensor 33 = Axis tripped during level sensing
10-11	2		Reserved

POSITION

Code	Class	Name	Length	Description
11 0x0B	2	POSITION	36 + (MAX_AXES *8)	Sends current positions for each of the axes on a board. This response is sent every 20 milliseconds when any axis movement is occurring or every 500 milliseconds when no axis movement is occurring Information on the status of axis enablement and limit switches is also returned. The packet also returns which motor control mode is active – (Position, Velocity or Independent) and whether or not home zeroing is in progress.

Position	Bytes	Format	Description
0	1	Character	Response Code = 11
1	1	Integer	Board number (optional according to application)
2	1		Home zeroing flag 0 = Home Zeroing not in progress 1 = Home zeroing in progress
3	1		Move mode (see pages 6 to 7) 0 = Velocity Mode 1 = Position Mode 2 = Independent Mode Note: When the controller issues a GOTO command, MRMC Robotics exits its previous mode (this could be either Velocity Mode or Position Mode)and enters Independent Mode. It remains in Independent Mode until the GOTO has completed execution. By observing the value of Move Mode, control software can see when a GOTO has completed execution.

4-5	2	Integer	Sequence Number Each POSITION command that is sent to the MRMC board arrives with a consecutive sequence number. The value returned here is the sequence number of the most recently received POSITION packet. This can be used to diagnose network difficulties.
6-7	2	Integer	Size 12 + 24 + (8 * AXES_PER_BOARD) e.g. 64 for 4 axis board e.g. 96 for 8 axis board
8-9	2	Integer	Interval The number of milliseconds that elapsed between the POSITION packet most recently received by the MRMC board and the packet previous to this one. This can be used to diagnose network difficulties.
10-11	2		Reserved
12-15	4	Integer	Number of ticks since the MRMC board was started (a tick is 1/50 of a second)
16-17	2	Integer	Interrupt number at which the MRMC board most recently received an IN_POSITION packet. When using Position Mode, the controlling software must send an IN_POSITION packet every 1/50 of a second. This time unit is known as a tick. MRMC_ROBOTICS has 32 interrupts for every tick (i.e. one interrupt every 625 microseconds). Thus, the arrival of each IN_POSITION packet is timed to an accuracy of 625 microseconds. This timing information is fed back as an interrupt number in the range 0 to 31.

			This information can be used to check the accuracy of the clock on the computer that is running the controller software.
18-19	2		Reserved
20	1	Character	Weather reporting control byte Bit 0 high = temperature & humidity reported Bit 1 high = excessive moisture reported Examples: 3 = all the above reported 0 = none of the above reported 2 = excessive moisture reported but temperature & humidity not reported
21	1	Character	Input Trigger status
22	1	Character	Any non-zero values = excessive moisture
23	1	Character	Lens-ID This is not yet implemented
24-25	2	Integer	Temperature in degrees centigrade
26-27	2	Integer	Percentage of relative humidity
28-31	8		Reserved
32-35	4	Long Integer	The time in milliseconds at which the most recently received POSITION packet was received by the MRMC board
36-39	4	Floating Point	Axis 1 Position
40-41	2		Axis 1 Status Information Bit 0

			A single bit reports whether or not the axis is "tripped" – i.e. whether MR_ROBOTICS is controlling the axis or whether an event has occurred which has caused MR_ROBOTICS to suspend control of the axis. Bit 0 = Tripped or not status of axis 1 Bits 1-3 3 limit switches exist for each axis. This allows for a limit switch at each extreme of travel of any device driven by the axis, and also for an additional "datum" switch that may be placed anywhere and used to re-calibrate a motor's exact position, without needing to move to either extreme of travel Single bits are used to report the status of each of these 3 limits individually Bit 1 = Limit 1 status of axis 1 Bit 2 = Datum (Limit 2) status of axis 1 Bit 3 = Limit 3 status of axis 1 Bits 9-4 = Axis Status Information A 6 bit code reports the axis status. 0 = operating normally 1 = limit switch tripped 2 = encoder out of position 3 = D2A overflow 4 = PWM overflow on servo motor 5 = Max stepper motor frequency exceeded 6 = tacho feedback error too large 7 = over current (certain devices only) 8 = over temperature (certain devices) 9 = axis disabled via DISABLE command 10 = CAN Bus error 11 = serially controlled lens disconnected . Bits 15 – 10 = Reserved
42-43	2		Reserved
44-51	1	8 bytes	Axis 2 Position & Status Information
52-end	1	8 bytes per axis	Position & Information for subsequent axes

READ NETWORK

Code	Class	Name	Length	Description
42 0x2A	3	READ NETWORK	56	This response is sent each time that MR_ROBOTICS receives either a READ NETWORK or a WRITE NETWORK command In the case of a READ NETWORK command, the values returned are the current values in flash memory. In the case of a WRITE NETWORK command, the response is sent on receipt of the command immediately BEFORE the IP address and associated network data is changed.

Position	Bytes	Format	Description
0	1	Character	Command Code = 42
1	1	8 bit integer	Board number (optional according to application)
2-3	2		Reserved
4-5	2		Reserved
6-7	2	16 bit integer	Size 56
8-11	4		Reserved
12-15	4	4 * 16 bit integer	1st byte of ip address e.g. 192 2nd byte of ip address e.g. 168 3rd byte of ip address e.g. 1 4th byte of ip address e.g. 236
16-17	2	16 bit integer	Maximum no. of axes that this board can control
18-19	2		Reserved
20-23	4	4 * 16 bit integer	1st byte of subnet mask e.g. 255 2nd byte of subnet mask e.g. 255 3rd byte of subnet mask e.g. 255 4th byte of subnet mask e.g. 0

24-27	4	4 * 16 bit integer	1st byte of default gateway e.g. 192 2nd byte of default gateway e.g. 168 3rd byte of default gateway e.g. 1 4th byte of default gateway e.g. 1
28-39	12	6 * 16 bit Integer	MAC address (not currently used)
40-55	16		Reserved

READ AXIS

Code	Class	Name	Length	Description
44 0x2C	3	READ AXIS	168	This response is sent each time that MR_ROBOTICS receives a READ AXIS command The values returned are the current values in flash memory. In the case of a READ AXIS command, the response will be immediate. In the case of a WRITE AXIS command, the response is sent once the new values have been written to Flash and then read back. This typically takes about 2 seconds.

Position	Bytes	Format	Description
0	1	Character	Command Code = 44
1	1	Integer	Board number (optional according to application)
2	1	Integer	Axis number to be configured – range 0-15
3	1		MEMORY TYPE 1 = FLASH_DIFFERS - Returning the axis setup data stored in flash memory. (SAVED AXIS DATA). It has been found that this data differs from the axis setup data currently stored 2 = FLASH_SAME - Returning the axis setup data stored in flash memory. (SAVED AXIS DATA). It has been found that this data IS identical to the axis setup data currently stored in RAM.

			3 = RAM - Returning the axis setup data currently stored in RAM (IN USE AXIS DATA).
4-5	2		Reserved
6-7	2	Integer	Size 104
8-11	4		Reserved
12-15	4		Reserved
16	1	Character	Motor Type 0 = Servo (External) 1 = Stepper 2 = Stepper with feedback 3 = DtoA 4 = DtoA with feedback 5 = Serially Controlled Iris 6 = Serially Controlled Focus 7 = Serially Controlled Zoom 8 = Direct Velocity 9 = CAN 10 = Robot 11 = Servo (PWM)
17	1	Character	Homing Style 1 = Search for limit 1 sensor (zero marker pulse available) 2 = Search for limit 2 sensor (zero marker pulse available) 3 = Search for limit 2 sensor (bi-directional) This method can be used with axes which rotate such as pan, tilt or turntable. It optimises homing time by optically detecting in which half of the travel the axis resides, and setting the sign of the homing velocity to be that which will reach the sensor soonest. 4 = Search for limit 1 sensor (no zero marker pulse available) 5 = Search for limit 2 sensor (no zero marker pulse available)

			6 = Search for limit 2 sensor (no zero marker pulse available) (bi-directional – see style 3 above) 7 = Lens Calibration Detects each end of travel of a lens 8 = As per 6, but, if on the sensor, does not check for the sensor for the first 10 seconds of searching. Required for standalone turntables. 9 = Slip Zero Moves at the homing velocity until the homing time has passed after which the axis moves to the position specified in the homing offset 10 = Direct The current position of the axis is set as zero 11 = Absolute Used when an axis is fitted with an absolute encoder and is thus able to report position without need of homing.
18	1	Character	Spare
19	1	Character	Goto Style 0 = All axes reach destination at same time 1 = NOT YET IMPLEMENTED (Each axis reaches its destination as swiftly as it can)
20	1	Character	Automatically Home on Start-up? 0 = No 1 = NOT YET IMPLEMENTED (Yes)
21-23	7		Reserved
24-27	4	Floating Point	Gearing ratio. This allows the controlling software to send position and velocity values in meaningful units such as degrees or inches etc.,

28-31	4	Floating Point	Maximum Acceleration
32-35	4	Floating Point	Maximum Velocity
36-39	4	Floating Point	Minimum Position NOT YET IMPLEMENTED
40-43	4	Floating Point	Maximum Position NOT YET IMPLEMENTED
44-47	4	Floating Point	Goto Speed Factor A value saying at what percentage of maximum possible speed, each goto should be executed.
48-51	4	Floating Point	Homing Velocity The velocity at which the axis seeks for its reference sensor
52-55	4	Floating Point	Homing Time Number of seconds allowed for the axis to seek for its reference sensor
56-59	4	Floating Point	Homing Offset After homing has calibrated the axis from the sensor, the axis can then be moved from that calibration position to a more convenient home position, offset by this value from the calibration position.
60-63	4	Floating Point	Backlash Offset This value is used where an axis has backlash. When an axis is moving in one direction, the specified offset is applied to compensate for the backlash. In the other direction, the backlash offset is not applied.

64-67	4	Integer	Signal gain (motor types SERVO or PWMSERVO)
68-71	4	Integer	Tacho gain (motor types SERVO or PWMSERVO)
72-75	4	Integer	Integral gain (motor types SERVO or PWMSERVO)
76-79	4	Integer	Current limit (motor types CAN, SERVO or PWMSERVO)
80-83	4	Integer	Temperature limit (motor type SERVO)
84-87	4	Integer	Positional Error Limit (all motor types)
88-91	4	Integer	Stepper pulse length (motor type STEPPER)
92-95	4	Integer	PWM type (motor types SERVO or PWMSERVO)
96-135	40	Floating Point	CAN Tuning parameters (motor type CAN)
136-167	32		Reserved

READ SETTING

Code	Class	Name	Length	Description
46 0x2E	3	READ SETTING	36 + (MAX_AXES * 4)	This response is sent each time that MR_ROBOTICS receives a READ SETTING command

Position	Bytes	Format	Description
0	1	Character	Command Code = 46
1	1	Integer	Board number (optional according to application)
2	1	Character	MEMORY TYPE When an MRMC ROBOTICS board is delivered from the factory it is preloaded with factory default axis settings. These settings are stored in flash memory and loaded into RAM when the board is powered up. Controlling software can alter these settings. This can be done either temporarily (altering the RAM copy of the setting using the LOAD SETTING command). Or permanently (altering the setting in Flash memory using the WRITE SETTING command). 0 = FROM FLASH – NO COPY TO RAM Returns the axis setting data stored in flash memory – doesn't copy this to RAM 1 = FLASH – COPY TO RAM Returns the axis setting data stored in flash memory & copies this into RAM 3 = FROM RAM Returns the axis setting data currently stored in RAM

4-5	2		Reserved
6-7	2	Size	Size 12 + 24 + (4 * AXES_PER_BOARD) e.g. 52 for 4 axis board e.g. 68 for 8 axis board
8-11	4		Reserved
12-35	24		Reserved
36 - 39	4	Where the value in bytes 2-3 of the READ SETTING command being responded to is between 11 and 30 32 bit float Otherwise 32 bit Integer	Setting value for first axis e.g. if Format is 1 then this is the motor type for the first axis e.g. if Format is 2 then this is the maximum acceleration for the first axis
40 - end		As per bytes 36-39 above	Setting value for each subsequent axis

READ METRICS

Code	Class	Name	Length	Description
51 0x33	3	READ METRICS	64	This response is sent whenever the controlling software requests measurement values (metrics) from a head which includes sensors for such values.

Position	Bytes	Format	Description
0	1	Character	Command Code = 51
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 104
8-11	4		Reserved
12-15	4	Floating Point	Accelerometer x
16-19	4	Floating Point	Accelerometer y
20-23	4	Floating Point	Accelerometer z
24-27	4	Floating Point	Gyro x
28-31	4	Floating Point	Gyro y
32-35	4	Floating Point	Gyro z
36-39	4	Floating Point	Magnetometer x
40-43	4	Floating Point	Magnetometer y
44-47	4	Floating Point	Magnetometer z
48-51	4	Floating Point	Pitch/Tilt Angle
52-55	4	Floating Point	Roll Angle
56-63	8		Spare

TEN PIN

Code	Class	Name	Length	Description
63 0x3F	3	TENPIN	48	This is a specialised packet returned to controllers that are authorised to control the 10-pin camera interface of a Nikon DSLR

Position	Bytes	Format	Description
0	1	Character	Command Code = 63
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 48
8-9	2	Integer	Status Code 0 = valid response data Non-zero = error codes
10-11	2		Reserved
12-47	36		Payload containing camera response

ZOOM TABLE ENABLED STATUS

Code	Class	Name	Length	Description
65 0x41	1	ZOOM TABLE ENABLED STATUS	14	Zoom Table Enabled Status This is sent back upon receipt of a GOTO command to indicate whether the zoom axis will be linearised during the upcoming GOTO. This is dependent on having received a valid zoom table, and on the GOTO start and end positions falling within the limits of the defined zoom table

Position	Bytes	Format	Description
0	1	Character	Command Code = 65
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 12
8-11	4	Integer	Reserved
12-13	2	Integer	0 = Zoom linerasiation will not occur during upcoming GOTO 1 = Zoom linearization will be employed during upcoming GOTO per existing table

VISCA

Code	Class	Name	Length	Description
67 0x43	3	VISCA	48	This is a specialised packet returned to controllers that controlling cameras via the VISCA camera control protocol

Position	Bytes	Format	Description
0	1	Character	Command Code = 67
1	1	Integer	Board number (optional according to application)
2-5	4		Reserved
6-7	2	Integer	Size 48
8-9	2	Integer	Status Code 0 = valid response data Non-zero = error codes
10-11	2		Reserved
12-47	36		Payload containing camera response

SOFTWARE UPDATES

Files Required

To update MR_ROBOTICS you need 2 files:

1. The new firmware itself
2. A pc executable file for sending the new firmware to the motor control board

The new firmware itself will have the suffix ***.btl***. Typical names are ***Hex_MHC.btl*** or ***Quad_Fed.btl.***

The name of the pc executable file is ***FirmwareUpdater.exe.***

Procedure for Updating

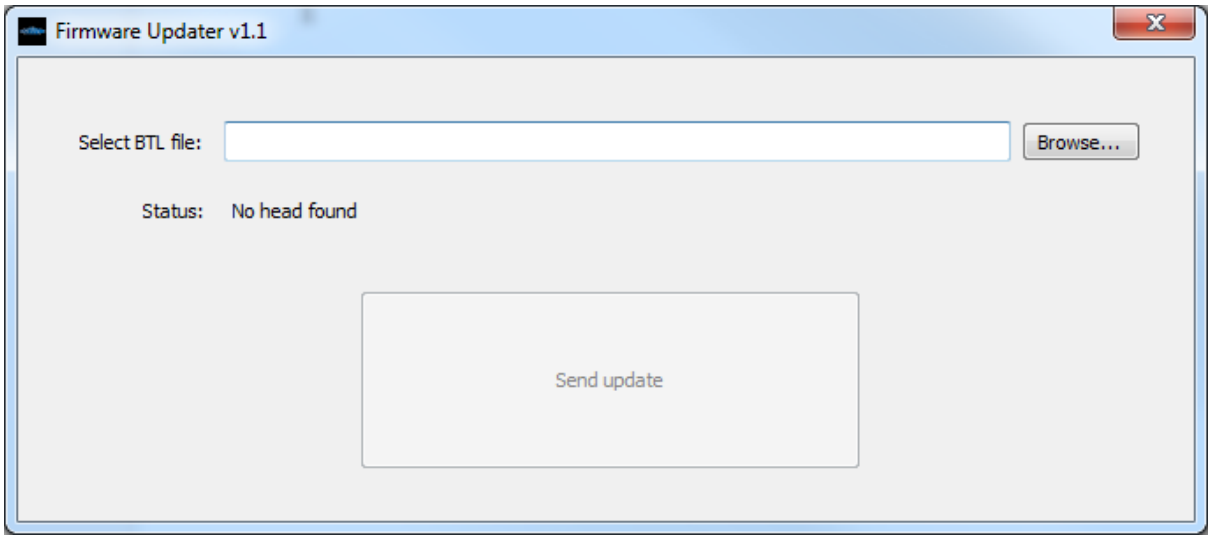
Follow the 8 steps below to update MRP_ROBOTICS.

1. Disconnect the motor control board to be updated from all networks.

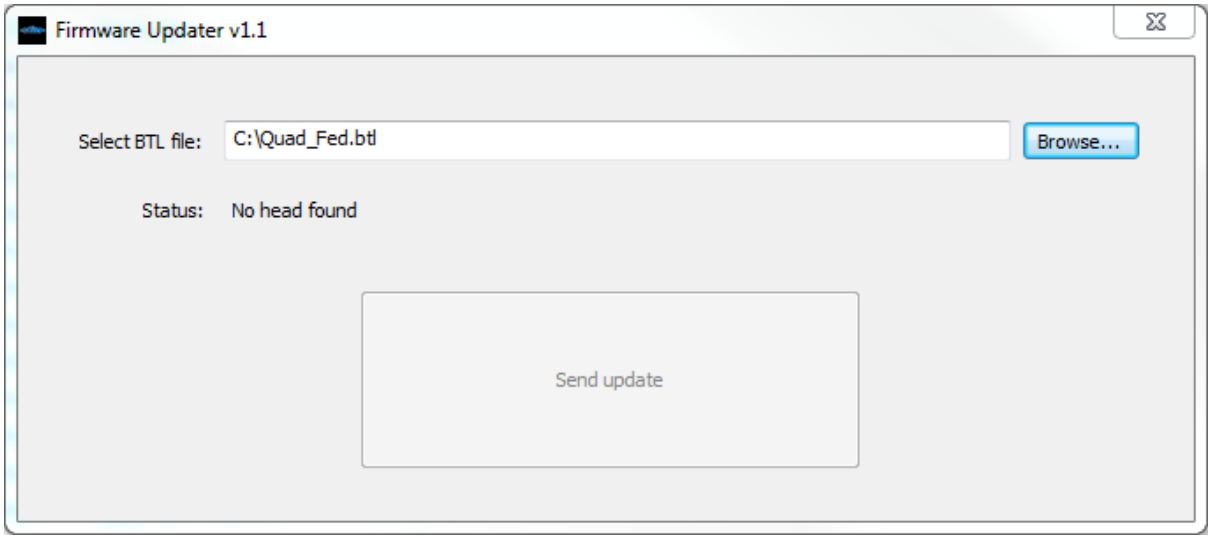
2. Turn off the power to the moto control board

3. Run an ethernet cable from the computer on which you are going to run ***FirmwareUpdater***, directly to the motor control board.

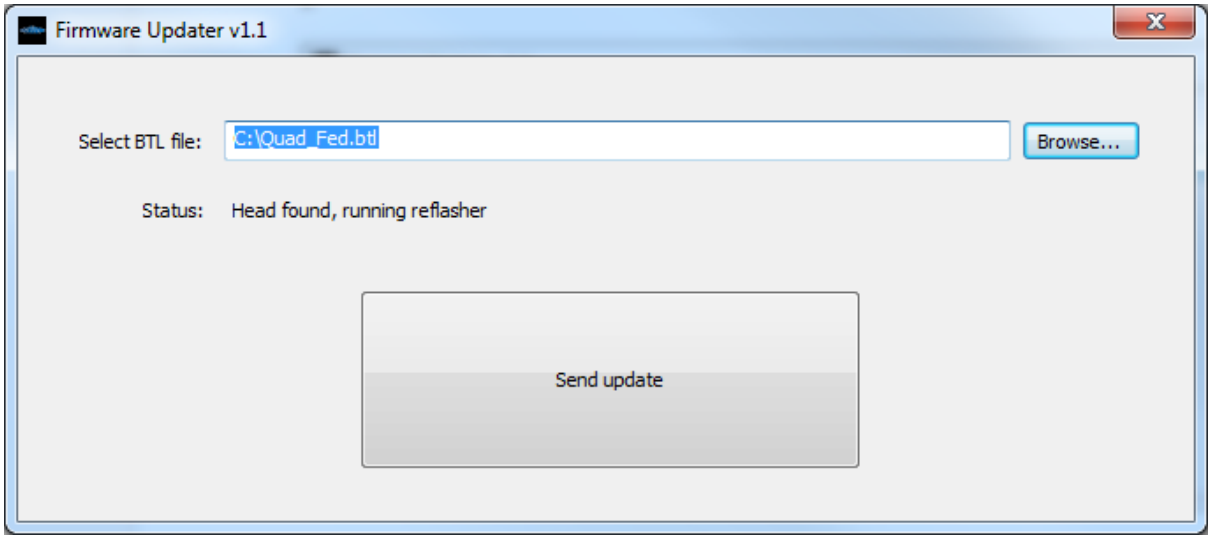
4. Start running FirmwareUpdater on your pc.



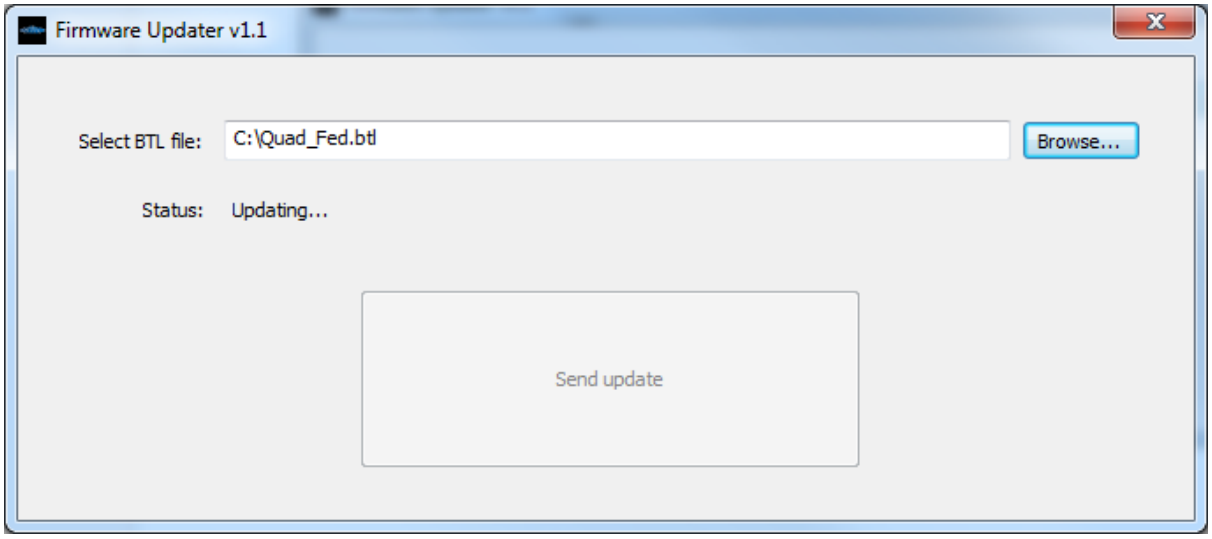
5. Select the firmware file you are going to upload to your head. This will have the suffix ***.btl***



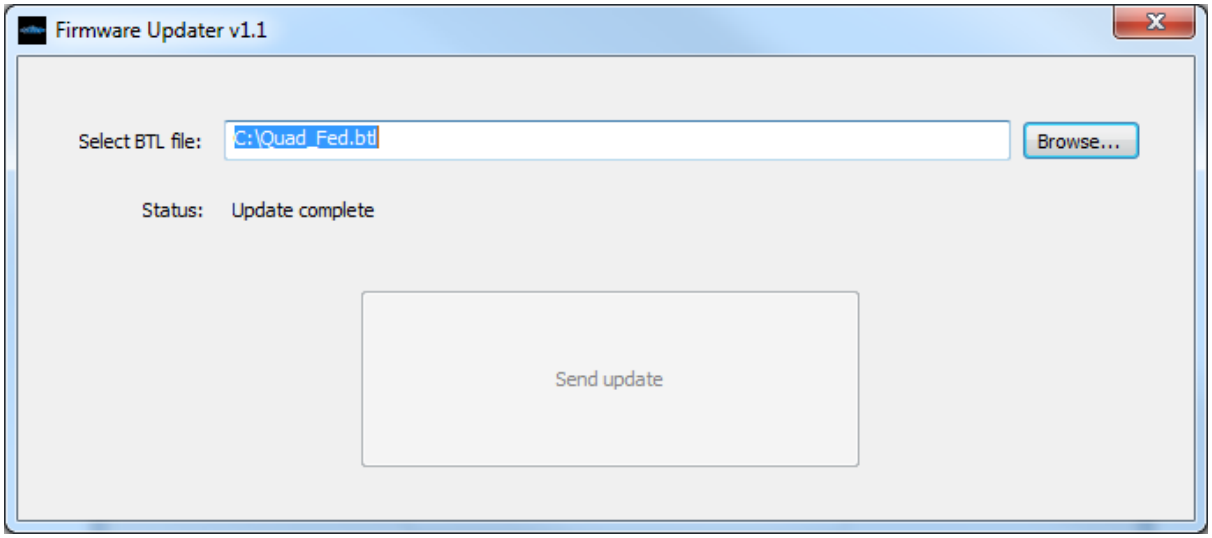
6. Power up the motor control board. After a couple of seconds or so, FirmwareUpdater will connect to the head. You will see the status message change from ***No Head Found*** to ***Head found, running Reflasher*** and the *Send update* button will become enabled. See below.



7. Press the Send update button. The status message changes to ***Updating***…



The firmware update now occurs. Once the update is complete, the Status message will change to ***Update complete***. This message confirms that your head has been successfully updated with the firmware update supplied.



8. Once you have seen the Status message change to *Update complete*, close ***FirmwareUpdater***.

Your head is now ready for use with the updated version of MR_ROBOTICS supplied.