



CAMMEGH Integration / Configuration Guide

CAMMEGH Roulette Wheel – Mercury Series

Mercury 360 (M360)

Mercury 360 RRC (M360RRC)

CAMMEGH Break Out Box (B.O.B.)



M360 Series

www.cammegh.com

1051 - INT - 02 - L

CAMMEGH
The World's Finest Roulette Wheel

About this guide

- This supplementary information has been developed for use with the models listed on the previous page only.
- This supplementary information is provided in addition to CAMMEGH M360 Series Owners Handbook, and is intended for software developers/integrators or experienced personnel use only.
- All Cammegh Roulette Wheels have been precision engineered to the finest tolerances for consistent non-biased operation in the gaming environment.
- Before performing any maintenance or calibration work, if in doubt, consult the M360 Series Owners handbook and appropriate wheel service booklet before attempting work.
- Further information on a topic is indicated by the symbols ➡📖 PN, and should also be consulted.
- Should you require further assistance, please contact us at support@cammegh.com

Contents

Section 1 - Serial Port Configurations and Communications Protocol

Section 2 - Software Updates

Section 3 - Configuring the Game Engine

Section 4 - Protocol Output

Section 5 - Error Handling

Section 6 – Bonus Number Output

Section 7 - Game Engine Configuration Command List

Section 8 – Mercury II Compatibility

© 2010 CAMMEGH LTD.

All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the written permission of CAMMEGH LTD.

The manufacturer reserves the right to modify, without prior notice, the technical specifications in order to accommodate the latest technical developments. CAMMEGH LTD. will provide information on the status of existing operating instructions and on any alterations and extensions that may be relevant.

Contents - Expanded

1	Communications	5
1.1	Communications Port Configuration	6
1.2	Communications Protocol - With Cammegh Break Out Box (B.O.B.)	6
2	Software Updates	6
2.1	Cammegh B.O.B. Software Update	6
2.1.1	Software for Updating Cammegh B.O.B.	7
2.1.2	Connection between PC and B.O.B.	7
2.1.3	Updating the B.O.B.	8
2.2	Cammegh GuzUnda Software Update	9
2.2.1	Software for Updating Cammegh GuzUnda	9
2.2.2	Connection between PC and GuzUnda via B.O.B.	9
2.2.3	Updating the GuzUnda	10
3	Configuring the Game Engine	11
3.1	Configuring Game Play	11
3.1.1	Game Modes	12
3.1.2	Game Parameters	13
4	Protocol Output	14
4.1	Configuring the B.O.B. Protocol Output	14
4.2	Cammegh Legacy Protocol Output	15
4.3	Winning Number Output	15
4.4	Basic protocol	16
4.4.1	Game Started	16
4.4.2	No More Bets	16
4.4.3	Winning Number	16
4.5	Advanced protocol	17
4.5.1	State	17
4.5.2	Game Count	17
4.5.3	Winning Number	17
4.5.4	Warning Flags	17
4.5.5	Rotor Speed	17
4.5.6	Rotor Direction	17
4.6	Extended Advanced protocol	18
4.6.1	Extended Flags	18
4.6.2	Spin Counter	18
4.6.3	Ball Direction	18
4.6.4	Ball Speed	19
4.6.5	Ball in Pocket Position	19
4.7	Game States	20
4.7.1	State 1 Start Game	20
4.7.2	State 2 Place Bets	20
4.7.3	State 3 Finish Betting	20
4.7.4	State 4 No More Bets	20
4.7.5	State 5 Winning Number	20
4.7.6	State 6 Table Closed	20
4.7.7	State 7 Dealer Lock	20
5	Error Handling	21
5.1	Error Flags	22
5.2	Error Flags Explained (Advanced Protocol)	22
5.2.1	Value 1 Rotor Speed	22
5.2.2	Value 2 Ball and Rotor Direction Match	22
5.2.3	Value 4 Sensor Failure	22
5.2.4	Value 8 Void Game	22
5.3	Respin	23

Contents - Expanded

6	Bonus Number Output	24
6.1	Cammegh Aurora Roulette	24
6.1.1	Aurora Prize Game	24
6.1.2	Aurora Win Colour/Number	24
6.2	Bonus Numbers 6 Random Number Generation	25
6.2.1	Bonus Numbers Bonus Number 1	25
6.2.2	Bonus Numbers Bonus Number 2	25
6.2.3	Bonus Numbers Bonus Number 3	25
6.2.4	Bonus Numbers Bonus Number 4	25
6.2.5	Bonus Numbers Bonus Number 5	25
6.2.6	Bonus Numbers Bonus Number 6	25
7	Addiitonal Commands	26
7.1	GuzUnda Game Log Commands	26
7.1.1	Access Event Log	26
7.1.2	Count of Log Entries	26
7.1.3	Delete Oldest Entry	26
7.1.4	Clear Entire Log	26
7.1.5	Find First Entry	26
7.1.6	Find First System Entry	26
7.1.7	Find First Status Entry	26
7.1.8	Find First Game Entry	26
7.1.9	Find Next Entry	26
7.2	Monitoring and Diagnostics Commands	27
7.2.1	Level Sensor	27
7.2.2	Pocket Position Diagnoistic	27
7.2.3	Rim Sensor Diagnostic	27
7.2.4	Report Firmware Version	27
7.2.5	Report X Axis Level	27
7.2.6	Report Y Axis Level	27
7.3	Custom Setup Commands	28
7.3.1	Set Rotor Custom Pockets	28
7.3.2	Set Mercury RRC mode	28
7.4	Value (Paramter) Commands	29
7.4.1	Query Value	29
7.4.2	Assign Value	29
7.4.3	Default Value	29
7.4.4	Value Indicies (Parameters)	29
8	Mercury II Compatibility	32
8.1	Mercury II Game Output Protocol	32
8.2	Mercury II Game Commands Comparison	32

Section 1 – Serial Port Configuration and Communications Protocol

1.1 – Communications Port Configuration

The Mercury 360 with Guzunda (on board game controller) can be connected directly via a 15pin serial to 9 pin serial converter with power supply outlet, or primarily via Cammegh BOB (Break Out Box) which has three serial ports configured as below, and ethernet connectivity.

Connector	Male 9-way Miniature 'D'
Pin Assignment	Identical to PC: Pin 2 = Receive; Pin 3 = Transmit; Pin 5 = Ground
Cable	To connect to a PC, use a Null Modem cable (3 wire female-to-female with pins 2, 3 and 5 connected, pins 2 and 3 being crossed over)
Baud Rate	9600
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None

1.2 – Communications Protocol – With Cammegh Break Out Box (B.O.B.)

For the first few seconds after power on the B.O.B. of the Mercury 360 will not respond and the status LED will be 'Red'; during this time the controller is initialising.

If the LED does not change to green there is a hardware fault with the controller. Attempt a hardware reset by removing the mains power connection and leaving for 30 seconds before connecting again. Should the indication LED not change to green after rest attempts, contact Cammegh.

All packets to and from the controller begin with an asterisk (0x2A) and end with carriage return (0x0D, \r) *and/or* line feed (0x0A, \n), except the winning number reports, which do not start with an asterisk for backwards compatibility with display devices.

The controller is insensitive to which of \r and \n are used, but always terminates its packets by carriage return *and* line feed, \r\n.

Section 2 – Software Updates

2.1 – Cammegh B.O.B. (Break Out Box) Software Update

Cammegh undertake a continuous improvement policy to provide the most robust solution to achieve performance and security in our products.

As part of the continuous improvement policy, Cammegh may develop software update versions over time, applicable for these products.

For the Cammegh BOB, the latest software version is:-

CAMMEGH BOB V1.45

The software version can be checked via a serial communications program connected to serial port 1 of the BOB by sending the *v <enter> command with a response

If unsure please contact us at support@cammegh.com

2.1.1 – Software for Updating Cammegh BOB

The following software is required to connect to the Cammegh BOB via Ethernet

FTP Transfer software such as FTPCommander (freeware)

- Install the FTP transfer program as per the instructions and load the program.

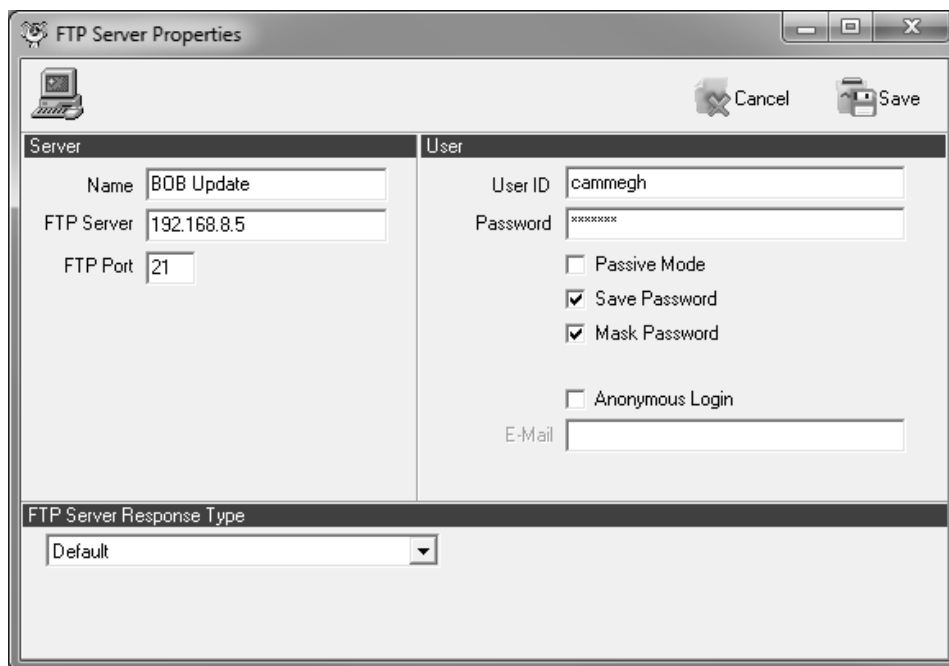
(Note:- the following instructions are based on use of the program ftpcommander.exe, setups may vary for other FTP transfer programs)



Section 2 – Software Updates

2.1.2 – Connection between PC and BOB

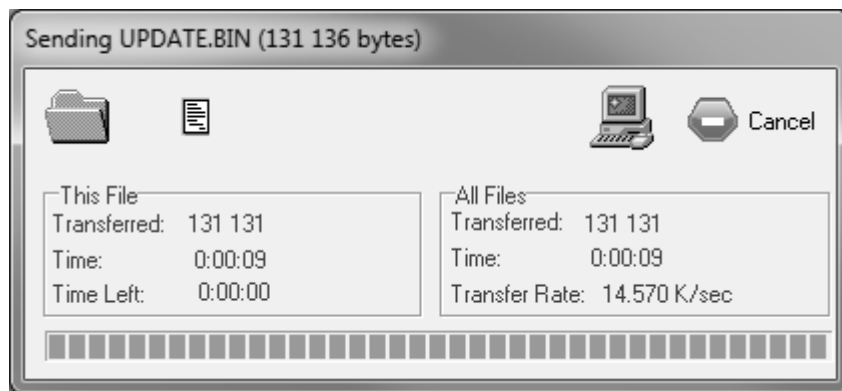
- Connect between the PC and the BOB Ethernet port using a standard RJ45 Ethernet cable.
- Connect between the PC and the BOB serial port 1 using a null modem serial 9 pin cable.
- Connect power to the BOB or if powered on, reset the BOB by removing the power connection and reconnecting after 10 seconds.
- Open a serial terminal program such as Hyperterminal, Realterm, or puTTY and establish connection to the BOB via the serial terminal, using the connection settings as per section 1.1 ➡ 5
- Enter the command ***ii <enter>** to retrieve the BOB IP address.
- In the FTP transfer program, select new server and enter the server name as 'BOB update'.
- Enter the IP address recovered from the BOB into the FTP Server field.
- Set the User ID to 'cammegh'.
- Set the password to 'hgemmac'.
- Ensure the "Passive mode" checkbox is CLEARED.
- Ensure the Save and Mask password checkboxes are SET.
- Press the "Save" button.



Section 2 – Software Updates

2.1.3 – Updating the BOB

- Unzip the supplied Cammegh BOB update file 'BOB_Version_X.XX.zip' using windows extraction agent and save the files to a directory on the PC hard-drive.
- In the FTP servers list, select the FTP server BOB Update and double click to establish connection between the PC and BOB (ignore error 502).
- Use the file tree of the FTPcommander program to select the director containing the Flash update file 'UPDATE.BIN'.
- In the Local Computer list, select the file 'UPDATE.BIN' and right click on the UPDATE.BIN file, selecting "Upload Selected Files".
- A transfer dialog should appear during the transfer, showing the transfer progress.



- DO NOT remove the power at any stage during the FW update process.
- The software version of the BOB can be checked by using the ***v <enter>** command in the serial terminal program.
- Enter the command ***m0 0 0 <enter>** in serial terminal program to reset the protocol mode.

Section 2 – Software Updates

2.2 – Cammegh GuzUnda (On-board Game Engine) Software Update

Cammegh undertake a continuous improvement policy to provide the most robust solution to achieve performance and security for our customers.

As part of the continuous improvement policy, Cammegh may develop software update versions over time applicable for these products.

For the Cammegh GuzUnda, the latest software version is:-

CAMMEGH GuzUnda V1.95

The software version can be checked via a serial communications program connected to serial port 1 of the BOB by sending the *V <enter> command with a response

2.2.1 – Software for Updating Cammegh GuzUnda

The following software is required to connect to the Cammegh GuzUnda via BOB serial port 1

FLASHProg_Version_1.18

- Install the flash programmer by using the windows extraction tool to extract FLASHProg_v1.14.exe

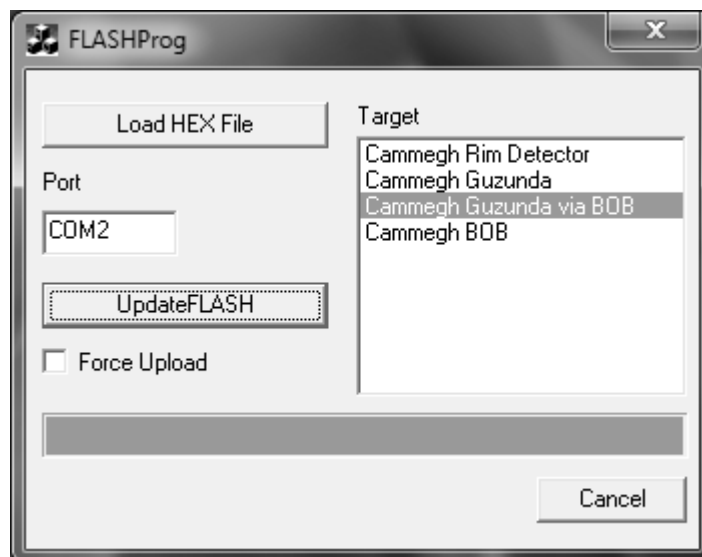
2.2.2 – Connection between PC and GuzUnda (via BOB)

- Connect between the PC and the BOB serial port 1 using a null modem serial 9 pin cable.
- Connect power to the BOB or if powered on, reset the BOB by removing the power connection and reconnecting after 10 seconds.

Section 2 – Software Updates

2.2.3 – Updating the GuzUnda

- Unzip the supplied Cammegh BOB update file 'GuzUnda_Version_X.XX.zip' using windows extraction agent and save the files to a directory on the PC hard-drive.
- Open the flash programmer program.
- Select "Cammegh GuzUnda via BOB" from the target list.
- Press the "Load HEX File" button
- Locate the folder containing the GuzUnda update and select "Update_Guzunda_vX.XX.hex"
- FLASHPROG should respond "Update File Loaded"
- Check the value in the comms port field is correct depending on the serial port connection of the PC
(! Note:- close any other open serial terminal programs using the comms port before attempting connection).
- **Remove the power supply from the BOB for 10 seconds, then replace.**
- **The 'Game State LED will show RED, then change to GREEN after a short period of time.**
- Press the "Update Flash" button **(Within 10 seconds of the game state LED changing GREEN)**
- A transfer dialog should appear during the transfer, showing the transfer progress.



- The transfer progress bar may stop or look to have stopped during the process, DO NOT remove the power or data connection, unless an error message is displayed.
- Upon reconnection, the BOB game state LED will be flashing Red. When connected and ready for use, the game state LED will be static Green.
- The software version of the GuzUnda can be checked by using the ***V <enter>** command in a serial terminal program.

Section 3 – Configuring the Game Engine

3.1 – Configuring Game Play

The Mercury 360 has been developed to offer the functionality game output of the Mercury II* series of in-rim sensor roulette wheels with additional security and performance benefits, producing an event log report per game played.

Within the GuzUnda game engine (GTI) there are a number of configurable modes and parameters to tailor game play to the individual's requirements.

The primary considerations and customisations are:-

- Selection of Game Mode – Mercury II compatible – The Cammegh 360 roulette wheel output is identical to the GAME OUTPUT of a Cammegh Mercury II series in-terms of gameplay and error warning flags, whilst creating an event log per game, logging rotor, ball, level and play events throughout each game. ➡📖 12

*Please take note of the Mercury II vs Mercury 360 compatibility table for commands, showing the difference between commands of the Mercury II and Mercury 360 variants. ➡📖 33
- Selection of Game Mode – Mercury 360 – The Cammegh Mercury 360 wheel output features an additional error warning flag (value 8) for ball spin errors, ball missing, or rotor tamper faults. ➡📖 12
- Setup of Game Parameters – No More Bets Event – The transition between state 3 (game started ball in rim) and state 4 (no more bets) is dependent on the speed of the ball reaching a set threshold. The threshold parameter can be adjusted to increase or decrease the no more bets period length. ➡📖 13
- Setup of Game Parameters – Ball Launch Speed – Mercury 360 mode only – The ball launch speed threshold can be adjusted to establish a 'bad' or 'weak' spin, constituting a void game. If the ball launch speed is below the threshold value, error 8 warning flag will be triggered and void the game, awaiting a respin or reset (depending upon Game Mode selected). ➡📖 13
- Setup of Game Parameters – Rotor Minimum and Maximum Speed – The rotor speed threshold can be adjusted to establish a 'suspicious' spin, triggering error 1 warning flag if the limits are exceeded. ➡📖 13

Section 3 – Configuring the Game Engine

3.1.1 – Game Modes


Cammegh Mercury 360 offers a number of game modes designed to offer flexibility in compatibility with third party devices using Mercury II game output, whilst still maintaining the benefits of the Mercury 360 security system.

To configure the game mode use the command:-

***G0=nn <enter>**

Where nn represents the game mode from the table below.

To access this setting a password is required *US351S <enter> (system unlock)

For more information regarding error warning flags, see section 5 – Error Handling ➡  22

Game Mode nn	Game Mode Description	Basic Protocol OP	Advanced Protocol OP	Extended Advanced OP	Direct Mercury II Compatible	Full Mercury 360 event log
00	Mercury II direct compatible mode (default)	✓	✓	✓	✓	✓
20	Mercury II compatible mode (*!) with no error 2 warning flag	✓	✓	✓	*!	✓
40	Mercury II compatible mode (*!) with no error 4 warning flag	✓	✓	✓	*!	*!
03	Mercury 360 mode with error 8 warning flag reporting for rotor errors (rotor index failure) only.	✓	✓	✓	✗	✓
1C	Mercury 360 mode with error 8 warning flag reporting for ball errors (ball direction change, missing, slow launch) only.	✓	✓	✓	✗	✓
1F	Full Mercury 360 mode with error 8 warning flag reporting for all rotor and ball errors.	✓	✓	✓	✗	✓
3F	Full Mercury 360 mode with error 8 warning flag reporting for rotor and ball errors, and error 8 instead of error 2 warning flag for ball in same direction as rotor spin direction.	✓	✓	✓	✗	✓

! Note: the game mode setting can be checked with the command ***G? <enter>**

*! Note: error will not display in error warning flag field.

Section 3 – Configuring the Game Engine

3.1.2 – Game Parameters

Cammegh Mercury 360 offers a number of game settings to tailor and fine tune the game play parameters to the customers individual gaming requirements.

To access these settings a password is required *US351S <enter> (system unlock)

To configure a game parameter use the command:-

#!xx=nnn <enter>

Where xx represents the game parameter , nnn represents the parameter value.


To check a game parameter value use the command:-


#?xx <enter>

Where xx represents the game parameter.

The typical game parameters fine tuned for game play can be found in the table below.

Game Parameter Value	Description	Default Value
66	No More Bets (State 4) transition based on Ball in rim RPM	040
6A	*Minimum Ball Launch Speed to start game (State 3) in RPM	050
A7	Transition from State 5 to new game State 1 in rotor revolutions	03 (hex value)

* ! Note 6A parameter will only instigate error 8 warning flag when playing in game mode 1C, 1F or 3C, i.e. Full Mercury 360 mode ➡  12.

For a full list of configurable game settings, see section 7 – Additional Commands ➡  29

To configure the game parameter values for the minimum and maximum rotor speed (error 1 warning flag), use the command:-

***Jnnn nnn <enter>**

Where nnn represents minimum rotor speed $1/10^{\text{th}}$ s RPM and nnn represents maximum rotor speed in $1/10^{\text{th}}$ s RPM

*E.G. *J100 300 gives a minimum rotor speed threshold of 10.0 RPM and maximum rotor speed threshold of 30.0 RPM for error 1 warning flag.*

! Note: the error 1 threshold setting can be checked with the command ***J? <enter>**

Section 4 – Protocol Output

4.1 – Configuring the Protocol Output of the BOB Serial Ports

The Cammegh GuzUnda game engine is capable of outputting three levels of game protocol for use with Cammegh ancillary devices such as Cammegh Billboard™, Pitboss HQ™, Pitboss Ti and Td Keypads™, Billboard Multiboard™, as well as third party devices such as betting terminals and gaming servers.

The three levels of protocol output are:-

Protocol Output (Legacy)	Protocol Mode value	Protocol Explanation
Cammegh Basic	1	➡ 📖 16
Cammegh Advanced	0	➡ 📖 17
Cammegh Extended Advanced	2	➡ 📖 18

For more information on the protocols, please see the relevant explanation page.

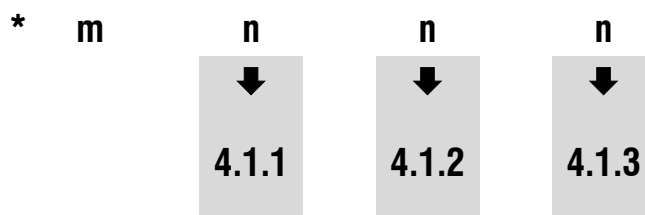
The Cammegh BOB features three serial communication ports and an Ethernet port. Each of the serial ports can be configured to output a different mode of protocol.

To configure the protocol output of the BOB serial ports:-

***m n n n <enter>**

Where n represents the protocol mode value, per BOB serial port.

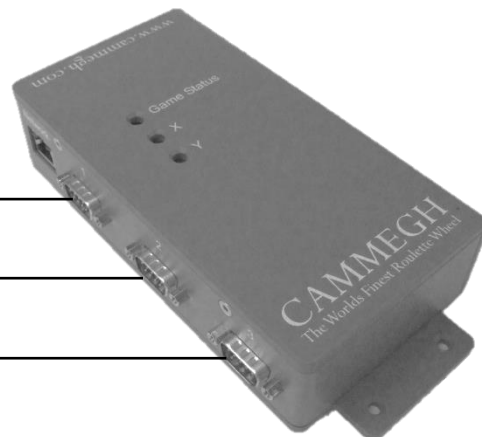
*E.G. *m 1 0 2 will give a BOB output configuration of protocol 1 (Cammegh Basic) from serial port 1, protocol 0 (Cammegh Advanced) from serial port 2, and protocol 2 (Cammegh Extended Advanced) from serial port 3.*



4.1.1 – Serial Port 1

4.1.2 – Serial Port 2

4.1.3 – Serial Port 3



Section 4 – Protocol Output

4.2 – Cammegh Legacy Protocol Output

The legacy protocol outputs game state information from the GTI (GuzUnda Game Engine) system every 500ms and when certain game events occur.

All legacy protocol lines are terminated with a “\r\n” pair

Rotor speeds below 2.0rpm are reported as 00.0rpm, speeds above 60.0rpm are reported as 00.0rpm.

4.3 - Winning Number output

“nn\r\n”

When the GTI system completes a game with a valid winning number the following alert is output to the legacy port(s). This happens regardless of the protocol mode selected.

The winning number always occupies two characters; ASCII characters 0-9.

The single zero wheel always outputs “0” for the zero pocket and the double zero wheel outputs “00” for the double zero pocket.

In the Basic protocol mode, the single winning number line is not output ➡  16

Section 4 – Protocol Output

4.4 - Basic Protocol – (1)

The basic protocol does not output a status line every 500ms.

The basic protocol outputs short messages to indicate the changes to the game states.

* S ➡ 4.4.1

* N ➡ 4.4.2

* W n n ➡ 4.4.3

4.4.1 - Game Started

“*S\r\n”

Sent to indicate entry into game started state, Start Game state, as described under Game States.

4.4.2 - No More Bets

“*N\r\n”

Sent to indicate entry into no more bets state, No More Bets state, as described under Game States.

4.4.3 - Winning Number

“*W nn\r\n”

Sent to indicate game completed with valid winning number, Winning Number state, as described under Game States.

Section 4 – Protocol Output

4.5 - Advanced Protocol – (0) (DEFAULT)

The advance protocol sends a status line every 500ms and no other messages other than the Winning No. output.

Each line has the format:

```
*  X  ;  n  ;  n  n  n  ;  n  n  ;  n  ;  n  n  n  ;  n
```

4.5.1 - State

1 character, ASCII number, decimal.

Indicates the current game state as described in Game States ➡ 20

4.5.2 - Game Count

3 characters, ASCII number, decimal.

The number of games played successfully. Range is 0 to 255 and wraps around. Internal game counter is 16 bits and counts up to 65535 games.

Game counter is reset on entry into Table Closed state.

4.5.3 - Winning Number

2 characters, ASCII number, decimal.

Last Winning Number in the same format as the Winning Number output. This field is set to double space if no winning number is known.

The Winning Number is cleared on entry into Table Closed state.

4.5.4 - Warning Flags ➡ 22

1 character, ASCII number, hexadecimal.

Warning flags related to the current game (value = sum of flags)

Bit 0 (value 0x1):

Wheel rotor is rotating at less than the minimum game limit, default 10.0rpm.

Bit 1 (value 0x2):

Ball has been launched in the same direction and the wheel rotor, GTI will not enter Winning No. state.

Bit 2 (value 0x4):

Rim Sensor error, object other than ball detected or sensor communications error. GTI will not give Winning No.

Bit 3 (value 0x8*):

Ball removed in Finish Betting or No More Bets states, or ball launched too slowly. GTI will not give Winning No.

* Note Error 8 Warning Flag output dependant on Game Mode ➡ 12

4.5.5 - Rotor Speed

3 characters, ASCII number, decimal scaled.

Rotor rpm value with one decimal place.

Multiplied by ten to remove the decimal point. Divide by ten to retrieve the actual rpm value.

4.5.6 - Rotor Direction

1 character, ASCII number, decimal.

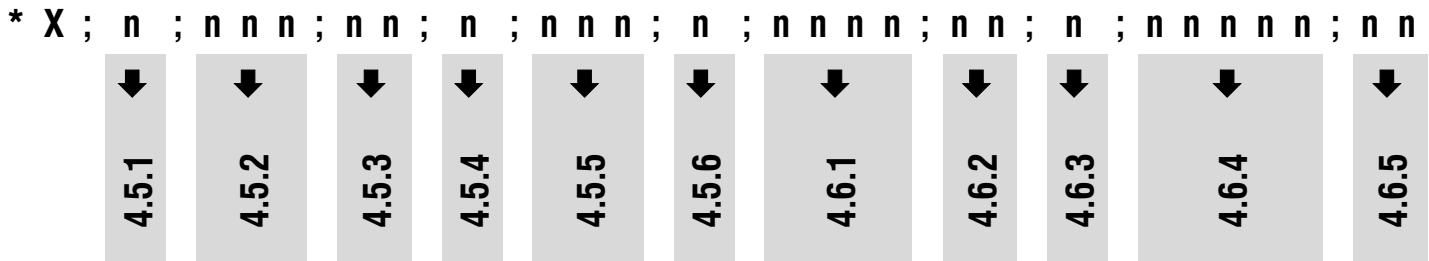
Rotor direction, 0=clockwise, 1=anti-clockwise.

Section 4 – Protocol Output

4.6 – Extended Advanced Protocol – (2)

The extended advance protocol sends a status line every 500ms and no other messages other than the Winning No. output.

Each line has the format:



4.6.1 - Extended Flags

4 characters, ASCII number, hexadecimal.

Status flags related to current game and system state (value = sum of flags)

Bit 0 (value 01):

Rotor is stopped.

Bit 1 (value 02):

Rotor is running at legal speed, i.e. with game minimum and maximum limits. ➡ 13

Bit 2 (value 04):

Rotor is running faster than game limit, default 30.0rpm. ➡ 13

Bit 3 (value 08):

Rotor is running slower than game limit, default 10.0rpm. ➡ 13

Bit 4 (value 0x10):

Rotor is rotating clockwise.

Bit 5 (value 0x20):

Rotor is rotating anti-clockwise.

Bit 6 (value 0x40):

Rotor direction is unknown.

Bit 7 (value 0x80):

No ball is detected in wheel.

Bit 8 (value 0x100):

Ball has been detected in pocket.

Bits 9-15 are unused and always zero.

4.6.2 - Spin Counter

2 characters, ASCII number, hexadecimal.

The number of times the ball has spun in the outer track in this game.

The spin counter value is reset to zero on entry into Place Bets state.

4.6.3 - Ball Direction

1 character, ASCII character, special.

Ball direction, '?'=direction unknown, 0=clockwise, 1=anti-clockwise.

Section 4 – Protocol Output

4.6 – Extended Advanced Protocol

4.6.4 - Ball Speed

5 characters, ASCII number, hexadecimal/special.

This value is the time the ball has taken to do 1 revolution of the wheel. If unknown it reports '?????'.

To calculate the time in seconds divide reported value in decimal by 32768.

Hence to calculate the ball speed in RPM:-

$\text{RPM} = 1966080 / \text{value in decimal}$

4.6.5 - Ball in Pocket Position

2 characters, ASCII number, special.

Ball in pocket position in the same format as the Winning Number output.

This field is set to '??' if no winning number is known.


Section 4 – Protocol Output

4.7 – Game States

The GTI system of the Cammegh GuzUnda contains a number of game states, which indicate the logical state of the game control engine. These states conform to the values used in the Mercury controller and are described below. The game state is also indicated by the game state tri-colour LED, changes to this LED are also detailed below.

4.7.1 - State 1: Start Game

This state is set when the wheel rotor starts rotating in Table Closed state or after the specified number of revolutions has passed since entry into Winning Number state, this defaults to 3 revolutions. The GAME STATE LED is set to GREEN.

The transition from state 5 to 1 is delayed by value A7 and can be checked using the command **#?A7** ➡  13

4.7.2 - State 2: Place Bets

This state is set when in Start Game state and there is no ball currently detected in the wheel. It takes at least one rotor revolution to detect this.

4.7.3 - State 3: Finish Betting

This state is set when a ball is detected in the outer track in Finish Betting state. It takes one revolution of the ball in the outer track to detect this.

If the ball is moving in the same direction as the rotor an error is recorded.

If using Mercury 360 mode and the ball launch is below the ball launch threshold value, an error is recorded.


If an error 2, or 8 warning flag is recorded, the game will not transition to state 4 without a respin.

4.7.4 - State 4: No More Bets

This state is set when ball is predicted to leave outer track in approximately 3.5 revolutions in Finish Betting state.

If the ball is moving in the same direction as the rotor an error is recorded.

The GAME STATE LED is set to RED.

The No More Bets event is triggered by value 66 and can be checked using the command **#?66** ➡  13

4.7.5 - State 5: Winning Number

This state is set when ball is detected in a valid pocket in No More Bets state and no other errors have occurred.

The GAME STATE LED is set to AMBER.

4.7.6 - State 6: Table Closed

This state is set when the wheel rotor has not moved for 10 minutes.

The GAME STATE LED is set to GREEN.

4.7.7 - State 7: Dealer Lock

This state is set when the dealer lock system is enabled and the GuzUnda is waiting for dealer acknowledgement before proceeding to Winning Number state.

Section 5 – Error Handling

5.1 – Error Flags

The GTI system of the Cammegh GuzUnda contains a number of error flags, dependant on which game mode and which protocol output is used.

Protocol Mode 0 – Cammegh Advanced

Error Flags details ➡  22.

Protocol Mode 1 – Cammegh Basic

No error flags

Protocol Mode 2 – Cammegh Extended Advanced

The Extended Advanced protocol output mode incorporates an additional error flag field, in addition to the error flag field as per the Advanced protocol output.

For more details of the extended error flags, ➡  18.

Section 5 – Error Handling

5.2 – Error Flags Explained (Cammegh Advanced Protocol)

The error warning flag field of the Advanced and Extended Advanced Protocol outputs uses the following values to indicate gaming errors (dependant on game mode selected, some warning flags may not be applicable). The field will contain the sum of the values (in hexadecimal) and normal operation will give a bit value of 0.

5.2.1 – Value 1 – Rotor speed is below or above the preset limits

- Limits are as determined by the command ***J** setting (Default 10-30RPM) ➡ 13
- This indicates that the rotor speed is outside of these limits.
- The warning flag will not stop a game playing out.
- The warning flag will be recorded in the event log for the game.

5.2.2 – Value 2 or Value A – Ball is travelling in the same direction as the rotor

- This indicates the game has been started with the ball spin in the same direction as the wheel is revolving.
- The warning flag will stop a game playing out and not transition from State 3 to State 4, unless the ball launch speed was below that of the No More Bets threshold parameter (command **#?66**) ➡ 13.
- A respin of the game will be required to continue play.
- The warning flag will be recorded in the event log for the game.

5.2.3 – Value 4 or C – Rim Sensor Failure/Error

- This indicates that two or more rim sensors have lost track of the ball or suffered a hardware or software malfunction, taking the rim sensors offline.
- The warning flag will stop a game playing out.
- The sensors will attempt automatic recover at all periods when the ball is sitting in a pocket and rotor revolving.
- If the 4 error remains following a reset of the game engine with a power off or *R command sent, there is a hardware fault with the sensors, please contact Cammegh support@cammegh.com.
- The warning flag will be recorded in the event log for the game.

5.2.4 – Value 8 – Void Game

- This indicates one or more of three scenarios have occurred to trigger a 'void' game.
 - The void game can be triggered if the ball has left the wheel and cannot be detected by the rim sensors.
 - The void game can also be triggered if the ball launch speed is below the parameter (command **#?6A**), i.e slow ball launch. ➡ 13
 - The void game can also be triggered if the rotor speed has suddenly changed indicating tampering of the rotor. (Only in Game Mode **1F** ➡ 12)

Section 5 – Error Handling

5.3 – Respin

The void game scenario will pause the game engine in state 3 or state 4 raising the warning flag. To clear the error flag and proceed with the game, a 'Respin' is required.

The following methods constitute a 'Respin' in the GTI system:-

1. The ball is placed in the last winning number pocket and allowed to revolve in the rotor for at least 2 revolutions. Before being re-spun either with or without a change of direction.
2. The rotor direction is changed and the ball is re-spun.
3. The rotor direction is changed, then changed to spin in the original direction and the ball re-spun.
4. A timeout of 10 seconds occurs from the ball leaving the wheel to the point where the ball is returned to the rotor or the rim.

The re-spin will reset the warning flag to a value of '0' and the game will continue.

If the error has not cleared, investigate the cause of the error flag further, consult the owners handbook, or contact support@cammegh.com

Section 6 – Bonus Number Output

6.1 – Bonus Number Output – Cammegh Aurora Roulette

The bonus number output for Cammegh Aurora roulette sends a status line at the following incidences:-

6.1.1 – Aurora Prize Game

At the start of State 3 (after the first protocol line in this state), a line in the format:-

```
"*Y:P:n\r\n"
```

Where n = 0 representing a normal (non-prize) game, or n=1 representing a prize game

6.1.2 – Aurora Winning Bonus Number

At the start of State 5 (after the first protocol line in this state and winning number line), a line in the format:-

```
"*Y:A:nn\r\n"
```

Where nn represents the winning bonus number in the range 00 to 07

When used with an M360 Wheel with Aurora Hardware, the bonus numbers relate to the following colours:-

- 00 = Rose
- 01 = Lilac
- 02 = Blue
- 03 = Sky Blue
- 04 = Lime Green
- 05 = Yellow
- 06 = Orange
- 07 = Fuchsia

For more information regarding Aurora roulette configurations, please contact Cammegh requesting document

ref: CAM MAN A360 GMG 01 – Aurora Roulette Game Mode Guide

Section 6 – Bonus Number Output

6.2 – Bonus Number Output – Cammegh Bonus Numbers 5 Random Number Generation

The bonus number output for Cammegh Bonus Numbers (5 RNG) sends a status line at the following incidences:-

6.2.1 – Bonus Numbers Bonus Number 1

At the start of State 3 (after the first protocol line in this state), a line in the format:-

“*Y:1:nnn\r\n”

Where nnn represents a bonus number (1) in the range 000 to 255

6.2.2 – Bonus Numbers Bonus Number 2

At the start of State 3 (after the bonus number 1 line), a line in the format:-

“*Y:2:nnn\r\n”

Where nnn represents a bonus number (2) in the range 000 to 255

6.2.3 – Bonus Numbers Bonus Number 3

At the start of State 4 (after the first protocol line in this state), a line in the format:-

“*Y:3:nnn\r\n”

Where nnn represents a bonus number (3) in the range 000 to 255

6.2.4 – Bonus Numbers Bonus Number 4

At the start of State 5 (after the first protocol line in this state and single winning number line), a line in the format:-

“*Y:4:nnn\r\n”

Where nnn represents a bonus number (4) in the range 000 to 255

6.2.5 – Bonus Numbers Bonus Number 5

At the start of State 5 (after the bonus number 4 line), a line in the format:-

“*Y:5:nnn\r\n”

Where nnn represents a bonus number (5) in the range 000 to 255

6.2.6 – Bonus Numbers Bonus Number 6

At the start of State 4 (after the bonus number 3 line), a line in the format:-

“*Y:6:nnn\r\n”

Where nnn represents a bonus number (6) in the range 000 to 255

Section 7 – Additional Commands

7.1 – GuzUnda Game Log Commands

7.1.1 - Access Event Log

Command “*Et\r\n”

Start an event log operation of the specified type ‘t’.

7.1.2 - Count of Log Entries

Command: “*E?\r\n”

Reply: “nnnnn\r\n”

Return the number of entries currently saved in the log

7.1.3 - Delete oldest entry

[Locked Command ‘L’]

Command: “*E-\r\n”

Reply: “OK\r\n”

Delete the oldest log entry.

7.1.4 - Clear entire log

[Locked Command ‘L’]

Command: “*E\$\r\n”

Reply: “OK\r\n”

Delete the entire log contents.

This command may take a few minutes.

7.1.5 - Find first entry

Command: “*EA\r\n”

Reply: “E <Event Data>\r\n” or “E None\r\n”

Find first entry of any type.

7.1.6 - Find first System entry

Command: “*EY\r\n”

Reply: “E <Event Data>\r\n” or “E None\r\n”

Find first entry of system type.

7.1.7 - Find first Status entry

Command: “*ES\r\n”

Reply: “E <Event Data>\r\n” or “E None\r\n”

Find first entry of status type.

7.1.8 - Find first Game entry

Command: “*EG\r\n”

Reply: “E <Event Data>\r\n” or “E None\r\n”

Find first entry of game type.

7.1.9 - Find next entry

Command: “*EN\r\n”

Reply: “E <Event Data>\r\n” or “E None\r\n”

Find next entry of the same type as the initial search.

Section 7 – Additional Commands

7.2 – Monitoring and Diagnostics Commands

7.2.1 - Level Sensor

Command: **"*SL\r\n"**

Reply: **"S OK\r\n"**

Show diagnostic for level sensor.

7.2.2 - Pocket Position Diagnostic

Command: **"*SP\r\n"**

Reply: **"SPd nn.n\r\n"**

Reply: **"S OK\r\n"**

Print the calculated pocket position each time a ball is detected by a rim sensor. Detector position is specified by (d). Ball position in pocket is number and pocket position fraction. I.e. 0.5 is middle of pocket.

7.2.3 - Rim Sensor Diagnostic

Command: **"*SD\r\n"**

Reply: **"SD <information>\r\n"**

Reply: **"S OK\r\n"**

Print diagnostic information related to the rim sensor operation.

7.2.4 - Report Firmware Version

Command: **"*V\r\n"**

Reply: **"V <version string>\r\n"**

Report the firmware name and version number.

7.2.5 - Report X and Y Axis Level

Command: **"*X\r\n"**

Reply: **"(X) nn.n (Y) nn.n\r\n"**

Report the current value of the X Axis level sensor.

Section 7 – Additional Commands

7.3 – Custom Setup Commands

7.3.1 - Set Rotor Custom Pockets

[Locked Command 'U']

Command: **"*Z nn,p1,p2...pn\r\n"**

Reply: "Z nn\r\n" or "Z INVALID\r\n"

Set the number of pockets on the rotor, uses a custom pocket order, specified using numbers p1-pn. Where there must be nn pocket numbers specified, 99 specifies a double zero, 00 specified single zero. The first pocket position is the zero index pocket and the pockets run anti-clockwise.

Up to 63 pockets can be supported.

7.3.2 - Set Mercury RRC mode (if fitted)

Command: **"*AH=n\r\n"**

Reply: "OK"

n = 0 = Mercury 360 mode

n = 1 = Mercury 360 RRC mode.

Section 7 – Additional Commands

7.4 – Value (Parameter) commands

User Configuration, System configuration and Calibration value can be queried and adjusted using the value command.

Values are accessed using a single character group identifier and a two-digit value index.

7.4.1 - Query Value

Command “#?nn\r\n”

Reply “# n..n\r\n”

Query the specified value (nn).

7.4.2 - Assign Value

Command “#!nn,n..n\r\n”

Reply “# OK\r\n”

Assign the specified value.

7.4.3 - Default Value

Command “#\$nn\r\n”

Reply “# OK\r\n”

Assign the specified value to its default value.

7.4.4 - Value indices (Parameters)

The tables on the following pages list some of the available value indices in the system.

For a full list contact Cammegh ref:- GTI –151-UM

Index	Type	Description	Group
00	WORD	Firmware Version	CAL
01	DWORD	GuzUnda Serial Number	CAL
02	WORD	Temperature Calibration (unused)	CAL
03	WORD	X Axis Calibration	CAL
04	WORD	Y Axis Calibration	CAL
05	BYTE	Hardware Flags	CAL
06	CHAR	Installed rim sensor count	CAL
07	DWORD	Rim Sensor 0 Serial Number	CAL
08	DWORD	Rim Sensor 1 Serial Number	CAL
09	DWORD	Rim Sensor 2 Serial Number	CAL
0A	DWORD	Rim Sensor 3 Serial Number	CAL
0B	DWORD	Rim Sensor 4 Serial Number	CAL
0C	DWORD	Rim Sensor 5 Serial Number	CAL
0D	INT	Rim Sensor 0 Clockwise position	CAL
0E	INT	Rim Sensor 1 Clockwise position	CAL
0F	INT	Rim Sensor 2 Clockwise position	CAL
10	INT	Rim Sensor 3 Clockwise position	CAL
11	INT	Rim Sensor 4 Clockwise position	CAL
12	INT	Rim Sensor 5 Clockwise position	CAL

Section 7 – Additional Commands

7.4.4 - Value indices (Parameters)

Index	Type	Description	Group
13	INT	Rim Sensor 0 Anti-Clockwise position	CAL
14	INT	Rim Sensor 1 Anti-Clockwise position	CAL
15	INT	Rim Sensor 2 Anti-Clockwise position	CAL
16	INT	Rim Sensor 3 Anti-Clockwise position	CAL
17	INT	Rim Sensor 4 Anti-Clockwise position	CAL
18	INT	Rim Sensor 5 Anti-Clockwise position	CAL
19	BYTE	Rim Sensor 0 Address Mapping	CAL
1A	BYTE	Rim Sensor 1 Address Mapping	CAL
1B	BYTE	Rim Sensor 2 Address Mapping	CAL
1C	BYTE	Rim Sensor 3 Address Mapping	CAL
1D	BYTE	Rim Sensor 4 Address Mapping	CAL
1E	BYTE	Rim Sensor 5 Address Mapping	CAL
1F	DWORD	Rotor Serial Number	CAL
20	BYTE	Rotor Inner Threshold	CAL
21	BYTE	Rotor Inner Ratio	CAL
24	BYTE	Rotor Inner Average	CAL
25	BYTE	Rotor Outer Threshold	CAL
60	BYTE	Rotor Timeout	CONST
61	BYTE	Rotor Minimum Period	CONST
62	CHAR	Rotor rpm Average	CONST
63	BYTE	Ball Landing Timeout Factor	CONST
64	BYTE	Ball Spinning Timeout Factor	CONST
65	CHAR	Ball Small NMB rpm Threshold	CONST
66	CHAR	Ball Large NMB rpm Threshold	CONST
67	BYTE	Ball Rim Factor	CONST
68	WORD	Ball Small Threshold	CONST
69	CHAR	Ball Small Slow rpm	CONST
6A	CHAR	Ball Large Slow rpm	CONST
6B	CHAR	Ball Re-spin rpm delta	CONST
6C	PIN	CAL PIN	CONST
6D	PIN	LOG PIN	CONST
6E	PIN	SYSTEM PIN	CONST
6F	PIN	USER PIN	CONST
70	PIN	MCB PIN	CONST
71	PIN	RIM SENSOR PIN	CONST
72	PIN	PROTOCOL LOCK PIN	CONST

Section 7 – Additional Commands

7.4.4 - Value indices (Parameters)

Index	Type	Description	Group
A0	BYTE	Configuration Flags	CONFIG
A1	WORD	Rotor Closed Timeout	CONFIG
A2	CHAR	Rotor Lowest rpm	CONFIG
A3	CHAR	Rotor Low rpm	CONFIG
A4	CHAR	Rotor High rpm	CONFIG
A5	CHAR	Rotor Highest rpm	CONFIG
A6	CHAR	Ball Settle count	CONFIG
A7	CHAR	Game WinNo revs	CONFIG
A8	CHAR	Level sensor clear threshold	CONFIG
A9	CHAR	Level sensor error threshold	CONFIG
AA	BYTE	MCB Flags	CONFIG
AB	CHAR	MCB Rotor speed	CONFIG
AC	CHAR	MCB Launch power	CONFIG
AD	CHAR	MCB Game rate	CONFIG
AE	BYTE	Custom Pocket Count	CONFIG
B0	BYTE	Custom Pocket table	CONFIG
EE	BYTE	Game Mode mask value	CONFIG

Abbreviations:-

BOB, Break Out Box.

Baud, communications data rate

\r, carriage return, ASCII code 0x0D, 13 decimal.

\n, line feed, ASCII code 0x0A, 10 decimal.


BYTE, value with 8 bit storage capacity

WORD, value with 16 bit storage capacity

DWORD, value with 32 bit storage capacity

Section 8 – Mercury II Compatibility

8.1 – Mercury II Game Output Protocol

The output of the Mercury 360 is identical to Mercury II when the game engine is configured in the default *G0=00 mode ➡  12.

This applies to the basic, advanced and extended advanced levels of protocol output.

8.2 – Mercury II Commands Comparison

The following page shows a table of Mercury II commands, the Mercury II output response, and the same command output for Mercury 360.

Please note the difference between the commands if integrating existing third party applications based on the original Mercury II platform. ➡  31

Section 7 – Mercury II Compatibility

7.2 – Mercury II Commands Comparison

M2 Command	Description	M2 Response	M360 Response	M360 Equivalent command
"*A\r\n"	Report the current wheel position as the number of pockets since the last 0 which have passed the datum point, in the direction returned by the *D command	"*A<space><ASCII number>\r\n"	"ERROR\r\n"	None
"*B\r\n"	Report the pocket which is currently in front of the in-rim reader	"*B<space><ASCII number>\r\n"	"ERROR\r\n"	None
"*C\r\n"	Calibrate the ball in pocket position	"*C <result>\r\n" 0=success	"LOCKED\r\n" or "INVALID\r\n"	"*CD\r\n"
"*D\r\n"	Report current wheel direction	"*D<space>0\r\n" clockwise or "*D<space>1\r\n" anticlockwise	"0\r\n" clockwise or "1\r\n" anticlockwise	Unchanged
"*J <slow rpm> <fast rpm>\r\n"	Setup the Extended wheel speed detection limits. Limits are RPM*10. E.g. default = *J 100 250\r\n (I.e. Slow = 10.0 RPM, Fast = 25.0 RPM)	None	"OK\r\n"	Unchanged
"*L\r\n"	Re-send last winning number	"*L<space><ASCII number>\r\n"	"*<ASCII number>\r\n"	Unchanged
"*M <end port> <middle port> <sensors>\r\n"	Configure the port protocols. 0 = advanced protocol, 1 = basic protocol, 2= Extended protocol. E.g. *M 0 0 2\r\n will set both ports to use the advanced protocol with a two sensor wheel.	Reply is the same as *V	"Locked\r\n"	"*m <port 1> <port 2> <port 3>\r\n"
"*O\r\n"	Report on the currently enabled optional features.	"*o<space><ASCII number>\r\n"	"ERROR\r\n"	None
"*o <feature number>\r\n"	Setup the currently enabled optional features. (Note you only have to do this once, features are stored even after power off)	"*o<space><ASCII number>\r\n"	"ERROR\r\n"	None
"*R\r\n"	Reset State back to state 1 (force game restart)	None	"OK\r\n"	Unchanged
"*T\r\n"	Report current wheel RPM	"*T<space><ASCII RPM*10>\r\n"	"OK\r\n"	None
"*V\r\n"	Report the version of the in-rim reader controller	"*V<version><ASCII string>\r\n" e.g. "*V3.22 (23-Apr-07) Cammegh In-Rim Reader, CC 3 Sensor (C) Syoptic Ltd\r\n"	"<ASCII string> <version>\r\n" e.g. "Cammegh GuzUnda 1.64\r\n"	Unchanged
"*Z <pockets>\r\n"	Setup the number of pockets on this wheel 37 (single zero) or 38 (double zero)	None	"OK\r\n"	Unchanged