Megasquirt serial protocol

Dated: 2014-10-28

This version of the documentation applies to:

- MS2, Microsquirt, Microsquirt-module, MSPNP2

  running MS2/Extra firmware 3.3.x or later

  OR
- MS3, MS3-Pro, MS3-Gold, MSPNP-Pro

  running MS3 firmware 1.2.x or later

  OR
- Microsquirt transmission control code version 0.019 or later

Does not apply to other Megasquirt products or other firmware versions.

Table of Contents

1 Introduction

This document covers the second generation error-checked Megasquirt serial protocol (aka 'newserial') in use on products since 2013. First the error correction 'wrapper' is described, then the pre-existing 'payload' data format. Prior products used a less robust serial protocol with no error checking or response codes.

Dashes or dataloggers that are intended to receive data only are encouraged to use CAN - the latest Megasquirt firmwares support 11bit CAN broadcasting for simpler integration. See the Megasquirt-CAN protocol document and accompanying .dbc file.

For compatability with existing third-party devices in the field, a subset of commands are supported without error checking. This is covered in section 6. New devices are encouraged to use the full error-checked protocol for serial data or CAN.

In the following sections, the various serial messages will be described. Each will have a legend showing the data similar to the following:

0	1	2	3	4	5	6	Byte number within message
Size		'A'	CRC32				Contents of those bytes.

Size is a two byte (16 bit) big-endian value in bytes 0 and 1.

'A' means the ASCII character uppercase A in byte 2.

CRC32 is a 4 byte (32 bit) big-endian value in bytes 3,4,5,6.

2 Overview

The Megasquirt serial protocol uses a request and response system. (Unless commanded, no serial data is sent from the Megasquirt - there is no broadcast realtime data stream.)

The comms are effectively half duplex - do not send data to the Megasquirt while it is transmitting.

Tuning device / laptop / dash : Request

0	1	2	3	4	..																							
Size		Payload: command and data as relevant........																							CRC32			

Megasquirt : Response

0	1	2	3	4	..																						
Size		Flag	Payload: command and data as relevant........																					CRC32			

For every packet sent to the Megasquirt, there will be a response packet, so the sender can validate correct reception. The sender MUST act on the response code and MUST check the CRC32.

3 Wrapper

The 'wrapper' describes the size prefix and CRC32 suffix to the serial payload.

Request format

0	1	2	3	4	..																							
Size		Payload: command and data as relevant........																							CRC32			

Size: This is the big-endian 16bit size of the packet including the two size bytes and four CRC32 bytes.

The maximum size depends on the Megasquirt hardware.

The 'f' command should be used to read this from the ECU and then break requests down into the appropriate sized blocks.

CRC32: This is the big-endian 32 bit CRC or all payload bytes i.e. excluding size and CRC. The CRC32 is implemented as per public domain 'crc32.c' (http://www.csbruce.com/software/crc32.c)

Response format

0	1	2	3	4	..																				
Size		Flag	Payload: command and data as relevant........																			CRC32			

The response is similar to the request but adds a single byte Flag. This flag is used to communicate the data type on a successful transaction or an error code.

Flag:

Flag	**Meaning**	**Comments**
0x00	OK	This is used in response to successful write commands.
0x01	realtime data	Successful response to 'A' with realtime data packet.
0x02	page data	Successful response to 'r' with tuning/configuration data.
0x03	config error	Alternate response to 'A' with configuration error message.
0x04	burn ok	Successful response to a 'b' command.
0x05	page10 ok	Special.
0x06	CAN data	Response contains data from remote device over CAN.
	Error codes	
0x80	underrun	There was a timeout before all data was received. (25ms per character.)
0x81	over-run	A byte (or bytes) were received before the serial interrupt was serviced.
0x82	CRC failure	CRC32 did not match
0x83	unrecognised command	The command within the payload was not recognised.
0x84	out of range	Address or page number or data size is not possible. Do not retry.
0x85	busy	Serial is already busy. Pause and retry.
0x86	flash locked	An attempt was made to write to sensor calibration tables while locked.
0x87	sequence failure 1	Special.

0x88	sequence failure 2	Special.
0x89	CAN queue full	No space to queue command to remote CAN device.
0x8a	CAN timeout	No response from remote CAN device.
0x8b	CAN failure	Other failure while communication with remote CAN device.
0x8c	parity error	Check your serial hardware.
0x8d	framing error	Check your serial hardware.
0x8e	serial noise	Check your serial hardware.
0x8f	txmode range	Data was received while a transmission was in process.
0x90	unknown serial error	Some other error condition.

In case of an error, the sender should take appropriate action. Some errors are transient and the command may be retried, while some indicate a permanent problem.

Each packet should be received in full and the CRC validated before any processing begins to prevent possible data corruption.

4 Payload

The payload format is largely the same as the original Megasquirt-2 serial protocol and supports communication with the local device and other devices on the Megasquirt CAN network. Each device has a "CAN id". The local device defaults to 0. For more information on Megasquirt CAN communications see the relevant manual.

All data is big-endian. (High byte first.) Data is sent in binary, there is no conversion to text, byte stuffing, alignment on words or escaping of characters.

4.1 Terms

4.1.1 CANid

The Megasquirt identifier of the device. The master ECU is always zero.

Other 'well known' ids

1 GPIO transmission controller

2 GPIO board

4 JBPerf TinyIOx

5 JBperf IO-x

7 Microsquirt transmission controller

4.1.2 Table

Within the Megasquirt memory map various regions are referred to as tables. This will vary depending on firmware revision and features. Consult the "ini" file supplied with the firmware as the final authority.

MS3 table list as per firmware 1.3.x

Table no.	Size	Internal name	Function

0	2048	cltfactor	Calibration table for CLT sensor.
1	2048	matfactor	Calibration table for MAT sensor.
2	1024	egofactor	Calibration table for AFR/EGO sensor.
3	2048	maffactor	Calibration table for MAF sensor.
4	1024	flash4	Tuning data. (TunerStudio 'page 1')
5	1024	flash5	Tuning data. (TunerStudio 'page 2')
6	-	canbuf	Used for CAN passthrough mainly.
7	varies	outpc / datax	Realtime data and data exchange.
8	1024	flash8	Tuning data. (TunerStudio 'page 3')
9	1024	flash9	Tuning data. (TunerStudio 'page 4')
10	1024	flash10	Tuning data. (TunerStudio 'page 5')
11	1024	flash11	Tuning data. (TunerStudio 'page 6')
12	1024	flash12	Tuning data. (TunerStudio 'page 7')
13	1024	flash13	Tuning data. (TunerStudio 'page 8')
14	60	Signature	Version and copyright string.
15	20	RevNum	Serial format string.
16	-	buf2	Special use.
17	1024	-	SDcard control.
18	1024	flash18	Tuning data. (TunerStudio 'page 9')
19	1024	flash19	Tuning data. (TunerStudio 'page 10')
20	2056	-	SDcard file readback.
21	1024	flash21	Tuning data. (TunerStudio 'page 11')
22	1024	flash22	Tuning data. (TunerStudio 'page 12')
23	1024	flash23	Tuning data. (TunerStudio 'page 13')
24	1024	flash24	Tuning data. (TunerStudio 'page 14')
25	1024	flash25	Tuning data. (TunerStudio 'page 15')
26	1024	trimpage	Read only data. (TunerStudio 'page 16')
27	1024	flash27	Tuning data. (TunerStudio 'page 17')
28	1024	flash28	Tuning data. (TunerStudio 'page 18')-
29	-	-	-
30	-	-	-
31	-	-	-
0xf0	1024	-	Tooth logger data.
0xf1	1024	-	Trigger logger data.
0xf2	1024	-	Composite logger data.
0xf3	1024	-	Sync error composite logger data.
0xf4	1024	-	MAP logger data.
0xf5	1024	-	MAF logger data.

0xf6	1024	-	Engine logger data.
0xf7	1024	-	Engine logger + MAP data.
0xf8	1024	-	Engine logger + MAF data.

Note that early Megasquirt firmwares only supported up to table 15. This means that you cannot use MS3 as a 'slave' device with one of the older firmwares (MS2/BG, MShift) as the pass-through master.

4.1.3 Offset

This is the address offset within a table, starting at 0. 16 bits big-endian.

4.1.4 Size

The number of bytes to read or write. 16 bits big-endian. Starting at 1 up to the maximum table size.

On Megasquirt-2 the maximum is 128 bytes, so to read a 1024 byte page a sequence of commands will be required. e.g.

read 128 bytes at offset 0

read 128 bytes at offset 128

read 128 bytes at offset 256 etc.

4.1.5 Serial version

The version number of this protocol. Presently 2.

4.1.6 Table blocking factor

The maximum size used to write to tables. Determine with 'f' command. (At time of writing, MS2 = 256, MS3 = 2048)

4.1.7 Write blocking factor

The maximum size used for general tuning data reads and writes. Determine with 'f' command. (At time of writing, MS2 = 256, MS3 = 2048)

5 Commands

5.1 'A' command

Returns realtime data.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6
Size		'A'	CRC32			

Megasquirt : Response

0	1	2	3	4	..																	
Size		Flag	Realtime data ...																			

CRC32			

5.2 'b' command

Burn tuning data to flash (make changes permanent.)

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8
Size		'b'	CANid	Table	CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

5.3 'c' command

Test serial communication. Responds with 16 bits seconds running.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6
Size		'c'	CRC32			

Megasquirt : Response

0	1	2	3	4	5	6	7	8
Size		Flag	Seconds		CRC32			

5.4 'f' command

For the selected CANid, it returns serial version, blocking factor for tables and blocking factor for writes.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7
Size		'f'	CANid	CRC32			

Megasquirt : Response

0	1	2	3	4	5	6	7	8	9	10	11
Size		Flag	Serial version	Table blocking factor		Write blocking factor		CRC32			

5.5 'F' command

Return serial version in ASCII e.g. currently 002.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6
Size		'F'	CRC32			

Megasquirt : Response

0	1	2	3	4	5	6	7	8	9
Size		Flag	'0'	'0'	'2'	CRC32			

5.6 'g' command (MS3 only)

Get selective outpc realtime data. (Returns error if not yet defined.)

This allows the tuning software to quickly fetch a defined dataset from the available realtime. The dataset must have been previously defined. See the "ini" file for the data required.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6
Size		'g'	CRC32			

Megasquirt : Response

0	1	2	3	4	..																	
Size		Flag	Selective realtime data ...																			

CRC32			

5.7 'h' command

Broadcasts a CAN 'halt' or 'unhalt' command to suspend non-essential CANbus usage. (Not fully supported.)

The message should also disable or enable CANbus usage on the local device. (Not currently implemented.)

Byte 3: 0 = unhalt

1 = halt

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7
Size		'h'	0/1	CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

5.8 'I' command

Returns the binary CANid of the directly connected device.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6
Size		'I'	CRC32			

Megasquirt : Response

0	1	2	3	4	5	6	7
Size		Flag	CANid	CRC32			

5.9 'k' command

Returns the CRC32 of a data page.

The offset and size fields are not used, set to zero.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'k'	CANid	Table	Offset = 0		Size = 0		CRC32			

Megasquirt : Response

0	1	2	3	4	5	6	7	8	9	10
Size		Flag	CRC of data page				CRC32			

5.10 'M' command

Returns the monitor version. (Used by firmware loader.)

Tuning device / laptop / dash : Request

0	1	2	10	11	12	13
Size		'M'	CRC32			

Megasquirt : Response

0	1	2	3	4	7	8	9	10
Size		Flag	Monitor version		CRC32			

5.11 'Q' command

Returns the serial format string. This should be used by tuning software to match to a serial format string in the "ini" file. The format string defines a particular tuning data format and realtime data format.

Tuning device / laptop / dash : Request

0	1	2	10	11	12	13
Size		'Q'	CRC32			

Megasquirt : Response

0	1	2	3	4	..										
Size		Flag	Serial format string (typically 20 bytes)												

CRC32			

5.12 'r' command

Read data from local or remote device.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	Table	Offset		Size		CRC32			

Megasquirt : Response

0	1	2	3	4	..										
Size		Flag	Data requested.												

CRC32			

5.13 'S' command

Returns the firmware version and copyright string. (The text that shows in the TunerStudio title bar.)

Tuning device / laptop / dash : Request

0	1	2	10	11	12	13
Size		'S'	CRC32			

Megasquirt : Response

0	1	2	3	4	..										
Size		Flag	Firmware version string (typically 60 bytes)												

CRC32			

5.14 'w' command

Write data to local or remote device.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	...	
Size		'w'	CAN id	Table	Offset		Size		Data to write...		

CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

6 Compatability mode

A number of the commands listed in section 4 may also be used without the wrapper in order to support older devices which are not aware of the current protocol.

As noted in the introduction, new devices are encouraged to use the error checked serial protocol or 11bit CAN protocols instead.

6.1 'a' command

Returns a subset of the realtime data formatted the same as MS2/BG firmware. See Appendix A for details.

Tuning device / laptop / dash : Request

0	1	2
'a'	0	6

Megasquirt : Response

0	1	2	..																
Realtime data ...																			

6.2 'A' command

Returns realtime data. The details are included in the ini file that matches the serial format string. A superset of the MS2/BG data.

Tuning device / laptop / dash : Request

0
'A'

Megasquirt : Response

0	1	2	..																
Realtime data ...																			

6.3 'c' command

Test serial communication. Responds with 16 bits seconds running.

Tuning device / laptop / dash : Request

0
'c'

Megasquirt : Response

0	1
Seconds	

6.4 'd' command

Developer use only. Enables debug buffer, if supported.

Tuning device / laptop / dash : Request

0
'd'

Megasquirt : Response

No response.

6.5 'D' command

Developer use only. Returns contents of debug buffer, if supported.

Tuning device / laptop / dash : Request

0
'D'

Megasquirt : Response

The contents of the debug buffer are returned.

6.6 'F' command

Return serial version in ASCII e.g. 001. (This is also supported as a compatability command.)

Tuning device / laptop / dash : Request

0
'F'

Megasquirt : Response

0	1	2
'0'	'0'	'1'

6.7 'I' command

Returns the binary CANid of the directly connected device.

Tuning device / laptop / dash : Request

0
'I'

Megasquirt : Response

0
CANid

6.8 'Q' command

Returns the serial format string.

Tuning device / laptop / dash : Request

0
'Q'

Megasquirt : Response

0	1	2	..																
Serial format string (typically 20 bytes)																			

6.9 'r' command

Read data from local or remote device.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6
'r'	CAN id	Table	Offset		Size	

Megasquirt : Response

0	1	2	..																
Data requested.																			

6.10 'S' command

Returns the firmware version and copyright string. (The text that shows in the TunerStudio title bar.)

Tuning device / laptop / dash : Request

0
'S'

Megasquirt : Response

0	1	2	..																
Firmware version string (typically 60 bytes)																			

7 Obsolete commands

The following commands were supported in some older Megasquirt firmwares, but are no longer supported.

7.1 'e' command

Was used to write and then read back data. Use 'w' and 'r' commands.

7.2 't' command

Was used to send calibration table data. Replaced with generic 'w' command.

7.3 'T' command

Was used to send calibration table data to a remote CAN device. Replaced with generic 'w' command.

7.4 'y' command

Was used to verify if ram and flash copies of data match.

8 SDcard serial protocol -(MS3)

Within the serial protocol, there are methods to communicate with and control the SDcard on the Megasquirt-3. This is not used at all on any Megasquirt-2 derivative.

All transfers (except noted) are wrapped in the newserial packet.

Table 0x11 will be used for bidirectional communication.

All numbers are big-endian.

8.1 SD do command (w 00 00)

Various control command. byte 9 sets the action.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13
Size		'w'	CAN id	0x11	0x0000		0x0001		XX	CRC32			

Where XX is:

00 Reset and return to normal

01 Reset and wait

02 Stop logging

03 Start logging

04 Put status into buffer

05 Re-initialise card

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

8.2 SD fetch buffer command (r 00 00)

Return XXXX bytes from buffer. Used by all read commands.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	0x11	0x0000		XXXX		CRC32			

Megasquirt : Response

0	1	2	3	4	..														
Size		Flag	Data requested.													CRC32			

8.3 SD status command

Requests long form status from SDcard system.

Note! Only use the status command when the card is already idle. Read outpc.sd_status first.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13
Size		'w'	CAN id	0x11	0x0000		0x0001		0x04	CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	0x11	0x0000		0x0010		CRC32			

Megasquirt : Response (16 bytesof payload)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18				
Size		Flag	X	Y	Sector size		Card size in sectors				No. files in root		Sector number of root directory				-		CRC32			

Byte X = Card status (same as outpc.sd_status)

bit 0: 0=No card, 1 =Card present

bit 1 : 0= SD, 1=SDHC

bit 2 : 0=Not Ready, 1=Ready

bit 3: 0=Not logging, 1=Logging

bit 4: 0=No error, 1=Error

bit 5: 0=V1.x, 1=V2.0 card

bit 6: 0=FAT16, 1=FAT32

bit 7: 0=normal, 1=not used

Byte Y = Error code

8.4 SD read directory command

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Size		'w'	CAN id	0x11	0x0001		0x0002		Directory chunk		CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	0x11	0x0000		0x0202		CRC32			

Megasquirt : Response

0	1	2	3	4	..														
Size		Flag	Data requested (514 bytes)													CRC32			

Returns: a sector with 32 bytes per file in root directory, plus U16 chunk number. Number within the payload:

Bytes 0-10 = 8.3 filename, space padded as per FAT directory

Byte 11 = 0=ignore, 1=file

Bytes 12-15 = undefined

Bytes 16-23 = absolute sector number (big endian)

Bytes 24-31 = file size _in_bytes_ (little endian direct from media)

Where the directory is longer than 32 entries, multiple reads will be required, chunk 0, 1, etc.

Note 1. The format is similar to the FAT16 directory structure, but MS3 returns sector number instead of cluster number. Non MS3 log files are ignored and not reported.

Note 2. All MS3 log files are created by MS3 as contiguous files. If these files are disturbed from the PC end and made non contiguous, data corruption on the SDcard will occur as the firmware does not support fragmentation due to the severe speed penalty it would incurr.

Note 3. directory chunk no. starts at 0.

8.5 SD read sector command

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	11	12	13	14
Size		'w'	CAN id	0x11	0x0002		0x0004		U32 sector number				CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	0x11	0x0000		0x0204		CRC32			

Megasquirt : Response

0	1	2	3	4	..										
Size		Flag	Data requested (512 bytes) + U32 sector number												

CRC32			

8.6 SD write sector command

The data sent is a full sector and then the 4 bytes of sector number.

Used incorrectly this command could corrupt the data on SDcard as it permits re-writing any area of the device (including MBR, FAT, directories etc.)

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8			
Size		'w'	CAN id	0x11	0x0003		0x0204		512 bytes of sector data		

9	10	11	12	11	12	13	14
U32 sector number				CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

8.7 SD read stream command

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13
Size		'w'	CAN id	0x11	0x0004		0x0001		0x01	CRC32			

Megasquirt : Response

A continual stream of 8bit data from the selected stream ADC input. Power cycle the MS3 to stop.

Note! Returned data is raw and not newserial packetised.

8.8 SD read compressed file command

The initial 'w' command sets up a large read command for the whole of a file. Successive 'r' commands are then used to read that compressed file back in 2k blocks. The 'w' command does not need to be repeated.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Size		'w'	CAN id	0x11	0x0005		0x0008		U32 sector number				U32 number of sectors total				CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

Then repeated 'r' commands with incrementing Block no. starting at zero.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	0x14	Block no.		0x0800		CRC32			

Megasquirt : Response

0	1	2	3	4	5	..									
Size		Flag	Block no.		Data requested (2048 bytes)										

CRC32			

8.9 SD erase file command

Erases a file on the SDcard. (MS3 will delete the directory entry and the FAT chain.)

A,B,C,D are space for the 4 byte file number in ACSII.

For filename LOG0002.MS3 send ascii '0' '0' '0' '2'. (48, 48, 48, 50)

Sending the actual directory block no. that the file entry appears in will speed up the deletion. Otherwise use zero to force code to find it.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Size		'w'	CAN id	0x11	0x0006		0x0006		A	B	C	D	Start dir block		CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

Busy bit in sd_status will be set during operation. Poll for completion.

8.10 SD speed test command

Send: w <canid> 11 00 07 00 04 <U32 sector number> <U32 num sectors>

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Size		'w'	CAN id	0x11	0x0007		0x0004		Sector number				Number of sectors to test				CRC32			

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

The code will blindy overwrite the sectors you request. Ensure there is no data there!

Poll until card is not busy.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	0x11	0x0000		0x000d		CRC32			

Megasquirt : Response

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Size		Flag	Sector down counter				Total time in 0.1ms units				Min time		Max time		X	CRC32			

The times can be used to calculate card speed and maximimum datalog rate.

X is status : 0 = running 1 = done 2 = error

8.11 RTC read command

Reads the local or CAN realtime clock

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12
Size		'r'	CANid	0x07	0x024d *		0x0008		CRC32			

* Address may change in future releases - consult ini file.

Megasquirt : Response

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Size		Flag	Sec	Min	Hr	dow	Date	Mon	Year		CRC32			

dow = day of week. 1 = Monday

8.12 RTC write command

Sets the realtime clock.

Tuning device / laptop / dash : Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Size		'w'	CAN id	0x07	0x027e *		0x0009		Sec	Min	Hr	dow	Date	Mon	Year		0x5a	CRC32			

* Address may change in future releases - consult ini file.

dow = day of week. 1 = Monday

Megasquirt : Response

0	1	2	3	4	5	6
Size		Flag	CRC32			

9 Appendix A - a 00 06 protocol

The compatability command a 00 06 allows simple reading of a subset of the realtime data. This is not recommended for new installs as you will miss out many fields that may be desirable.

The following lists the returned 112 bytes of data:

Offset	Size	Sign?	Name	Function	Units	Mult	Divide	Add
0	2	N	seconds	Seconds ECU has been on	s	1	1	0
2	2	N	pulseWidth1	Main pulsewidth bank 1	ms	1	1000	0
4	2	N	pulseWidth2	Main pulsewidth bank 2	ms	1	1000	0
6	2	N	rpm	Engine RPM	RPM	1	1	0
8	2	Y	advance	Final ignition spark advance	deg BTDC	1	10	0
10	1	N	squirt	Bitfield of batch fire injector events	-	1	1	0
11	1	N	engine	Bitfield of engine status	-	1	1	0
12	1	N	afrtgt1	Bank 1 AFR target	AFR	1	10	0
13	1	N	afrtgt2	Bank 2 AFR target	AFR	1	10	0
14	1	N	wbo2_en1	not used*	-	1	1	0
15	1	N	wbo2_en2	not used*	-	1	1	0
16	2	Y	barometer	Barometric pressure	kPa	1	10	0
18	2	Y	map	Manifold air pressure	kPa	1	10	0
20	2	Y	mat	Manifold air temperature	deg F	1	10	0
22	2	Y	coolant	Coolant temperature	deg F	1	10	0
24	2	Y	tps	Throttle position	%	1	10	0
26	2	Y	batteryVoltage	Battery voltage	V	1	10	0
28	2	Y	afr1	AFR1	AFR	1	10	0
30	2	Y	afr2	AFR2	AFR	1	10	0
32	2	Y	knock	Indication of knock input	%	1	10	0
34	2	Y	egocor1	EGO bank 1 correction	%	1	10	0
36	2	Y	egocor2	EGO bank2 correction	%	1	10	0
38	2	Y	aircor	Air density correction	%	1	10	0
40	2	Y	warmcor	Warmup correction	%	1	10	0

42	2	Y	accelEnrich	TPS-based acceleration	%	1	10	0
44	2	Y	tpsfuelcut	TPS-based fuel cut	%	1	10	0
46	2	Y	baroCorrection	Barometric fuel correction	%	1	10	0
48	2	Y	gammaEnrich	Total fuel correction	%	1	10	0
50	2	Y	ve1	VE value table/bank 1	%	1	10	0
52	2	Y	ve2	VE value table/bank 2	%	1	10	0
54	2	Y	iacstep	Stepper idle step number or PWM idle value duty	step duty%	1 392	1 1000	0 0
56	2	Y	cold_adv_deg	Cold advance	deg	1	10	0
58	2	Y	TPSdot	Rate of change of TPS	%/s	1	10	0
60	2	Y	MAPdot	Rate of change of MAP	kPa/s	1	10	0
62	2	Y	dwell	Main ignition dwell	ms	1	10	0
64	2	Y	MAF	Mass Air Flow (Scaling depend on range, 650g/s shown)	g/s	1	100	0
66	1	N	fuelload	'Load' used for fuel table lookup e.g. equals MAP in Speed-Density	%	1	10	0
68	2	Y	fuelcor	Adjustment to fuel from Flex	%	1	1	0
70	1	N	portStatus	On/off outputs status bits.	-	1	1	0
71	1	N	knockRetard	Ignition retard due to knock	deg	1	10	0
72	2	Y	EAEfcor1	Fuel correction due to X-Tau or EAE 1	%	1	1	0
74	2	Y	egoV1	Voltage from O2#1	V	1	100	0
76	2	Y	egoV2	Voltage from O2#2	V	1	100	0
78	2	Y	amcUpdates	not used*	-	1	1	0
80	2	Y	kpaix	not used*	-	1	1	0
82	2	Y	EAEfcor2	Fuel correction due to X-Tau or EAE 2	%	1	1	0
84	2	Y	spare1	not used*	-	1	1	0
86	2	Y	spare2	not used*	-	1	1	0
88	2	Y	trig_fix	not used*	-	1	1	0
90	2	Y	spare4	not used*	-	1	1	0

92	2	Y	spare5	not used*	-	1	1	0
94	2	Y	spare6	not used*	-	1	1	0
96	2	Y	spare7	not used*	-	1	1	0
98	2	Y	spare8	not used*	-	1	1	0
100	2	Y	spare9	not used*	-	1	1	0
102	2	Y	spare10	not used*	-	1	1	0
104	2	N	tachCount	not used*	-	1	1	0
106	1	N	ospare	not used*	-	1	1	0
107	1	N	cksum	not used*	-	1	1	0
108	4	N	deltaT	not used*	-	1	1	0

* The fields marked "not used" may be used in some alternate Megasquirt firmwares.

10 'ini' file

The Megasquirt firmwares ship with .ini file that is used by the tuning software. This describes the full serial data interface - both the calibration data and the realtime live data. The relatime data is a super-set of the data described in section 9.

This section will give a very brief introduction to understand how to read that section of the ini file.

Extract of ms3.ini file, from Megasquirt-3 firmware 1.4 :

```
[OutputChannels]
   deadValue        = { 0 } ; Convenient unchanging value.
   ochBlockSize     = 507 ; change this if adding extra data to outpc
```

ini file section start

in this firmware version, there are 507 bytes

```
#if CAN_COMMANDS
   ochGetCommand    = "r\$tsCanId\x07%2o%2c" ; leave this alone
```

Code to build serial command.
e.g. r 00 07 00 00 01 fb

```
#else
; fast get via serial
   ochGetCommand       = "A"
#endif
```

Alternate 'A' command to read all data.

```
   scatteredOffsetArray = qfrtfielddata
   scatteredOchGetCommand = "g"
   scatteredGetEnabled =  { scatterRuntimeEnabled && (tsLocalCanId ==
tsCanId) }
```

Definitions of 'quick' realtime data fetch command 'G'

```
   seconds          = scalar, U16,    0, "s",   1.000, 0.0
#if PW_4X
   pulseWidth1      = scalar, U16,    2, "ms",   0.004, 0.0
   pulseWidth2      = scalar, U16,    4, "ms",   0.004, 0.0
#else
   pulseWidth1      = scalar, U16,    2, "ms",   0.001, 0.0
```

Data fields.
Field name
=
scalar or bits,
size (U = unsigned, S = signed, 8 = 8 bits, 16 = 16bits),
Offset within dataset,
"Units",
Multiply raw number by,
Offset to add to raw number

```
   pulseWidth2      = scalar, U16,    4, "ms",   0.001, 0.0
#endif
   rpm              = scalar, U16,    6, "RPM", 1.000, 0.0
   advance          = scalar, S16,    8, "deg", 0.100, 0.0
   squirt           = scalar, U08,   10, "bit", 1.000, 0.0
   ; Squirt Event Scheduling Variables - bit fields for "squirt" variable above
   ; inj1:    equ    3       ; 0 = no squirt         1 = squirt
   ; inj2:    equ    5       ; 0 = no squirt         1 = squirt
   ; sched1:  equ    2       ; 0 = nothing scheduled 1 = scheduled to squirt
   ; firing1: equ    0       ; 0 = not squirting     1 = squirting
   ; sched2:  equ    4
   ; firing2: equ    1
   firing1          = bits,   U08,   10, [0:0]
   firing2          = bits,   U08,   10, [1:1]
   sched1           = bits,   U08,   10, [2:2]
   inj1             = bits,   U08,   10, [3:3]
   sched2           = bits,   U08,   10, [4:4]
   inj2             = bits,   U08,   10, [5:5]
```

The #if PW_4X allows an alternate scaling when rarely used setting PW_4X is enabled.

In normal mode, a pulsewidth raw number of 12345 is converted to 12.345ms

firing1 etc. illustrate bitfields
i.e. firing1 is bit 0
firing 2 is bit 1
etc.

The exact data size will vary with firmware version as features are added. Once a firmware becomes "release" the size should be stable. The 'Q' command is used to query the serial format string.

Near the top of the ini file, there is a line similar to:

```
   signature      = "MS3 Format 0513.03 " ; MS-II sends a null at 20th byte
```

This should be used to match an ini file to a particular ECU firmware.