CAN Communications Protocol

For MGL Avionics CAN bus devices

Release 10 16th of August 2018

MGL CAN Bus
MGL CAN bus devices operate using standard CAN bus protocol at 250KBaud. Please note that no devices include any bus termination resistors. These are to be fitted externally. For typical, short run connections, it is quite reasonable to install a single resistor at the host side of a value of 60 to 120 ohms. The CAN bus cannot work without this resistor. Standard CAN wiring demands a twisted pair cable run with a 120 ohm resistor terminating each side. Devices are connected to this bus using short stubs.

Multiple devices on one CAN Bus
Multiple CAN devices can be connected to a single CAN bus.

Each CAN devices must have a unique address on the bus. MGL Devices have a fixed address or address range in case more than one device of the same type is to be connected (for example RDACs).
In case of multiple devices of the same type addresses within the devices range are assigned either by dip switches or via a specific command on the CAN bus (this is for example used on the AHRS devices).

In case of CAN bus servos you the process involves connecting a single servo to the CAN bus and then sending the identification message containing the desired servo number (1-16).
Once the servo has been identified, it retains this identification for future use until changed by a new identification message.

MGL CAN Addressing
Addresses are 11 bits and identify the message source. Lower 4 bits are used as message type ID, upper 7 bits are device address where a single node may have more than one device address to identify logical function blocks.

In this document, “CAN address” of a device refers to the upper 7 bits of the native CAN address. The lower 4 bits is the “ID” or function of the message allowing up to 16 message types to be sent to each device.

The Host logical device shall be device address 1 to 15. Address 0 is reserved for future use.
Currently, only host address 1 is utilized. Other host addresses are reserved and will be used as the number of different CAN messages needed to be sent to various devices increases. In a typical system, there is only one physical host active at any one time (more than one may be connected but only one will address devices, I.e. send data). Devices never send data directly to another device as this would imply two (or more) nodes on the CAN bus potentially using the same CAN address in a message and this is not allowed. However devices can of course receive data from other devices when it is transmitted.

Servos will occupy addresses 16-31. The address for the Compass / SP6 is the SP6 number - 1+36. The address for the AHRS / SP7 is the SP7 number – 1+40 and so on.

The complete address range for all current MGL CAN bus devices:

Master/Host: 1-15 (this range of addresses are used by the EFIS or other host to send messages to devices).
Servos: 16 – 31 (selected by software command)
RDAC: 32 – 35 (max 4 RDAC units per bus – selected by dipswitch)
Compass: 36 – 39 (max 4 Compass units per bus, selected by software command)
AHRS: 40 – 43 (max 4 AHRS units per bus, selected by software command)
Transponder 44 -- 45 (max 2 transponders per bus, selected by software command)
Reserved: 46 – 47
SP10: 48 – 51 (4 addresses determined by function – selected by dipswitch)
ECB: 52 – 59 (up to 8 x 8 breakers, selected by dipswitch)
TouchPad: 60 – 63 (up to 4 touch pads, selected by software command)
EFIS Extender: 64-67 (up to 4 extenders, selected by software command)
External MGL AP: 68 – 69 (up to two external AP systems)

RDAC XF uses two dip switches to set the CAN address.
SEL
B A
off off – Address 32
off on – Address 33
on off – Address 34
on on – Address 35

RDAC are always assigned addresses from 32. If one RDAC is used, address 32 is used. If two RDACs are used, address 32 and 33 are used and so on...

ECB uses DIP switches 1,2,3 to set CAN bus address from 52 to 59 (8 address slots).

Messages from Host (EFIS) to devices

Message: Host Sets SP-6/7/9 Device ID Number				
Message ID: 5			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	CONST	0xAA	-	Set Code
1	CONST	0x55	-	-
2	Byte	32 ... 43	-	SP6/7/9 number (1...4)
3	Byte	Byte 2 XOR 0xFF	-	-

Activates the device ID as specified in data Byte 2. Example: if this messages is sent to Device ID 37, the device will broadcast Compass 2 data. Each device can only transmit one group of data – in other words one device will not report data as Compass 1 and Compass 2. Some devices (example SP-7) can report multiple groups of data. In the case of the SP-7 it can report Compass and AHRS data.

Sending a number outside of the valid range has no effect.

Message: Host Broadcasts Aircraft Speed & Attitude				
Message ID: 2			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	sint	-1800-1800	0.1 deg	AHRS Bank angle
2...3	sint	-900-900	0.1 deg	AHRS Pitch angle
4...5	sint	0..3599	0.1 deg	AHRS YAW angle
6...7	word	0...1000	mph	Ground Speed (TAS if No GPS)

This message is sent at a rate higher or same to 1 per 500mS. It contains either TAS or GPS ground speed as determined by the connected system.

If AHRS angles are not known, values are to be set to $7FFF. These angles are mainly used by the SP-6 to compensate the magnetometer readings for the aircraft's attitude.

Message: Command to SP7/SP9 AHRS				
Message ID: 3			**Data length:** 1-7	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0-255		Command
1...7				Optional data

Only one command is defined at this point.:
0: Set Gyro derived horizon to that given by the accelerometers. No data.

Message: Command to SP6 Compass				
Message ID: 4			**Data length:** 1-6	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0-255		Command
1	byte	0-255		Command (repeat)
2...7				Optional data

Commands defined for pre-self calibrating compass firmware
0: Set North
1: Set South
2: Set East
3: Set West
4: Clear alignment
5: Start deviation calibration
6: End deviation mode and keep result
7: Cancel deviation mode (does not keep result)
8: Restore factory settings (clears above calibrations)
9: Operation mode. Note: only values 2 and 4 are supported = ACEL 3D and EFIS 3D

Note: EFIS 3D mode uses the EFIS to calculate alignments and 3D compensation. In this mode only raw data is of relevance to the EFIS.

None of the commands has any data. Note: Ensure data length of CAN packet is 2 bytes. SP-6 will reject any command message that is not 2 bytes in data size.

Commands defined for self-calibrating compass firmware

12: Clear calibration and start a new calibration
13: End calibration and store result. If sent without a calibration active will restore calibration from last stored.
14: Set pitch and bank angles to zero (assumes SP-6 is level)
15: Set bank angle to 90 degrees (assumes 90 degree bank, zero degrees pitch)
16: Set pitch angle to 90 degrees (assumes 90 degree pitch, zero degrees bank)

Note: Direction of physical pitch and bank for 90 degrees is not relevant.

Message: Set SP6 Compass bus address				
Message ID: 5			**Data length:** 4	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0xAA		key
1	byte	0x55		key
2	byte	36-39		Requested CAN address
3	byte	Byte 2 xor 0xFF		Requested CAN address (bits flipped)

This message should be sent with one SP-6 connected to the CAN bus. After receiving

this message the SP-6 will assume the requested address. This is intended to allow multiple SP-6 devices in a system.

Message: Command to Motor control unit (SP-10)				
Message ID: 8			**Data length:** 1-8	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0-255		Command
1	byte	0-255		Command (repeat)
2...7				Optional data

This sends a message to a Flap/Trim/Motor controller. Note: The function of the controller unit is programmed by means of a DIP switch array in the controller itself. Messages sent must match the function of the controller. Multiple controllers with different function assignments can be used on the same CAN address.

Commands defined:
0: Set position, Flap motor.
Position 0-4095, two bytes LSB first.
Motor direction: 0 or 1
Timeout: 0-255. Steps of 100mS. 0=No timeout.
Power: 0-255
1: Set position, Pitch trim. Data as above.
2: Set position, Bank trim. Data as above.
3: Reserved
4: Pulse Flap motor
Time: 0-255 for 0-2.55 seconds
Direction: 0 or 1
Power: 0-255
5: Pulse Pitch trim motor
Time: 0-255 for 0-2.55 seconds
Direction: 0 or 1
Power: 0-255
6: Pulse Bank trim motor
Time: 0-255 for 0-2.55 seconds
Direction: 0 or 1
Power: 0-255
7: Reserved
8: Setup. D2 is setup item, optional data follows at D3
0 – Flaps up position 0-4095. D3 LSB ,D4 MSB
1 – Flaps mid1 position 0-4095. D3 LSB ,D4 MSB
2 – Flaps mid2 position 0-4095. D3 LSB ,D4 MSB
3 – Flaps down position 0-4095. D3 LSB ,D4 MSB
4 – Pitch up position 0-4095. D3 LSB ,D4 MSB
5 – Pitch neutral position 0-4095. D3 LSB ,D4 MSB
6 – Pitch down position 0-4095. D3 LSB ,D4 MSB
7 – Roll right position 0-4095. D3 LSB ,D4 MSB
8 – Roll neutral position 0-4095. D3 LSB ,D4 MSB
9 – Roll left position 0-4095. D3 LSB ,D4 MSB
10 – Flaps motor timeout. 0-255. D3

11 – Flaps motor force. 0-255. D3
12 – Flaps motor DIR. 0-255. D3
13 – Pitch motor timeout. 0-255. D3
14 – Pitch motor force. 0-255. D3
15 – Pitch motor DIR. 0-255. D3
16 – Roll motor timeout. 0-255. D3
17 – Roll motor force. 0-255. D3
18 – Roll motor DIR. 0-255. D3
19 – Request setup data

Note: If any setup message is received, it will trigger a transmission of setup data identical to setup request 19.

9: Set to stored position

D2=0 Flaps to position in D3 – 0,1,2,3 = up,mid1,mid2,down
D2=1 Pitch to position in D3 – 0,1,2 = up,neutral,down
D2=2 Roll to position in D3 – 0,1,2 = right,neutral,left

Note: This message is usually only used for Flaps.

10: Request Diagnostics (RAW ADC values)

Message: General Command				
Message ID: 9				**Data length:** 1-8
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0-255		Destination
1	byte	0-255		Command
2...7				Optional data

Used for mode or calibration settings and other general commands to devices.

This message exists in RDAC firmware from 29th October 2013 onwards
Destination 0-3 is RDAC 1,2,3,4
Destination 8-15 is ECB 1,2,3,4,5,6,7,8

If destination is RDAC:
Command:
$81: Request calibration data
$82: Set Ambient calibration in following two bytes
$83: Set TC calibration in following two bytes
$84: Set analog calibration in following two bytes
$85: Set MAP calibration in following two bytes
$86: Set volt meter calibration in following two bytes
$A0: Write calibration to flash memory (store permanent)

If destination is ECB:
Command:
$00: Set breaker current trip level in steps of 0.1A.
Data[2] = breaker ID, 0 = breaker 1
Data[3],Data[4] = Trip current (LSB/MSB)
$01: Set breaker trip speed in steps of 0.1 seconds. Range 0 – 2.0 seconds.
Data[2] = breaker ID, 0 = breaker 1
Data[3] = trip time

$02: WIGWAG state. If one or two breakers are configured as WIG/WAG then this command witches the breaker into and out of WIGWAG mode.
Data[2] = 0 – WIGWAG off. 1 – WIGWAG on.
$03: WIGWAG time.
Data[2] = WIGWAG time in 0.1S steps. Default is 0.5 seconds = value 5.
$04: Breaker ON/OFF
For systems that are have hardwired “ON” breakers, this command switches breakers ON and OFF. It is also used to reset a tripped breaker by attempting to switch it ON.
Data[2] = breaker ID, 0 = breaker 1
Data[3] = 0 - breaker OFF, 1 – breaker ON (Reset).
$05: Request setup.
Sends setup packets as described under ECB to EFIS messages.

Note: 8 breakers per ECB module. Numbered 0-7. If breakers are paralleled, then switching either of the doubled breakers automatically switches the other as well. Current limits apply for total current in case of a doubled breaker. The ECB will halve the received trip current setting and apply this to each of the doubled breakers. It is OK to send the total trip current for just one of the doubled breakers.
Breakers can be doubled, tripled and quaded.
Trip times specify the trip time at trip current level. At higher levels, trip times are automatically reduced so to simulate a traditional breakers behaviour. In case of doubled breakers, it is OK to send the trip time for just one of them (the same time will be applied to the other).

ECB operating details are mostly selected by means of the DIPSWITCH array on the breaker itself.

Message: EFIS Extender command				
Message ID: 10,11,12,13			**Data length:** variable	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte			Type
1...7	byte			Data depends on type

Type 0 – set outputs. Place output value as bit pattern in bits 0..4 of byte 1. Length of message 2 bytes. “1” means switch output on (open collector output switched to ground).

Type 1,2,3,4,5 – Setup RS232 port 1,2,3,4,5

Bytes 1,2,3 – LSB first: Desired baud rate
Byte 4 – Bits 0..1 =Stop bits 0=0.5 1=1 2=1.5 3=2
Bits 2..3 =Parity 0=none, 1=even, 2=odd
Bits 4..7 =Data bits. (use “8”, only 8 bits supported)
Byte 5 – RX Timeout in increments of 250uS, 63.750mS max timeout = 255
Byte 6 – Threshold, number of bytes in buffer before TX of this data (unless timeout)

The port will receive data. If the timeout value since the last byte has been exceeded the data in the buffer will be sent (if any). If the threshold (number of bytes) has been exceeded, data will also be sent. It is recommended to use a threshold of 8 as this is the maximum size of data payload in a CAN package.

Type 6 – Set address

Place the value 0,1,2 or 3 in byte 1 and the same value XOR'ed with 0xFF in byte 2 to set the CAN sub address for this extender. This will set the CAN device address to 64,65,66 or 67. This implies that only one extender module may be connected at any one time as the addresses are set. This setting is stored in non-volatile memory and applies immediately.

Type 7,8,9,10,11 – TX data to a RS232 port

Place the data to be transmitted in bytes 1 to 7. This implies you can send between 1 and 7 bytes per CAN message.

Note: The extender module has a buffer of 256 bytes for each RS232 port and 1024 bytes for each RX port. Please keep this in mind, if you send more data as can be sent by the port given the ports baudrate and buffer size you will cause data to be lost.

Note: The CAN messages for the Extender documented here are never used in a MGL EFIS environment as the extender will connect to an EFIS using a dedicated high speed serial link. These messages are documented here for third party use. The CAN ports on the extender module are normally addressed via the high speed serial link and send and receive data as per normal EFIS operation. CAN addresses have nevertheless been chosen to be compatible with the MGL CAN system even if this is not strictly required.

If the Extender receives message type 0 (“set outputs”) addressed to it on any of its CAN ports (addresses 64-67) then it immediately switches into a mode where both CAN ports are dedicated ONLY to the messages documented here and the high speed data link is disabled. If this message is not sent, then the Extender will not send any of the messages documented or react on any of these messages. The default behaviour passes any CAN messages received to and from the high speed data link to the EFIS.

Message ID 10,11,12,13 corresponds to addressing extender CAN address 64,65,66,67

Messages from devices to Host (EFIS)

SP-7 Accelerometer Data

The SP-7 will transmit the Accelerometer Data message 20 times per second. This is a default message format that all Stratomaster instruments expect. This message will be sent unconditionally.

Message: SP-7 (AHRS) Broadcasts Accelerometer Data				
Message ID: 1			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	int	- 32000... +32000	1/1000 G	Scaled Accelerometer X
2...3	int	- 32000... +32000	1/1000 G	Scaled Accelerometer Y
4...5	int	- 32000... +32000	1/1000 G	Scaled Accelerometer Z
6...7	int	-32000... +32000	1/1000 G	Total G

SP-7 Gyro Rate

The SP-7 will transmit the Gyro Heading and Turn Rate message 20 times per second. This is a default message format that all Stratomaster instruments expect. This message will be sent unconditionally.

Message: SP-7 (AHRS) Gyro rates				
Message ID: 2			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	sint	-60...60	0.1deg/m	Turn Rate
2...3	sint	-32768-32768	*	Bank Rate
4...5	byte	-32768-32768	*	Pitch Rate
6...7	version	-32768-32768	*	Yaw Rate

*) Bank, Pitch and Yaw rates are scaled such that 360 degrees/second = 16384 giving a resolution of 0.022 degrees/second

SP-6 Heading and Raw Magnetometer Data

The SP-6 will transmit the Heading & Raw Magnetometer Data message 20 times per second. This is a default message format that all Stratomaster instruments expect. This message will be sent unconditionally.

Raw mag readings are unsigned 12 bit numbers with the value 2048 equal to zero field strength. Numbers are little endian and packed.

Message: SP-6 (Compass) Broadcasts Heading & Raw Magnetometer Data

Message ID: 1				**Data length:** 8 bytes
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	12 bits	0...35999	1/100deg	Magnetic Heading
2...3.5	12 bits	0...4095	-	Corrected Raw Magnetometer X
3.5...5.0	12 bits	0...4095	-	Corrected Raw Magnetometer Y
5.0...6.5	12 bits	0...4095	-	Corrected Raw Magnetometer Z
7	byte	-128..127		Slip angle (ball position)

SP-6 Data sent during active deviation calibration

The SP-6 will send three CAN packets at intervals during calibration. Note: These messages shown here are created in pre-self calibrating firmware versions. For newer versions please see the next section.

Message: SP-6 (Compass) Deviation compensation data packet 1				
Message ID: 2				**Data length:** 8 bytes
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	word	0...65536	-	Maximum EW reading
2...3	word	0...65535	-	Minimum EW reading
4...5	word	0...65535	-	Maximum NS reading
6...7	word	0...65535	-	Minimum NS reading

Message: SP-6 (Compass) Deviation compensation data packet 2				
Message ID: 3				**Data length:** 8 bytes
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	word	0...65536	-	EW current reading
2...3	word	0...65535	-	NS current reading
4...5	word	0...65535	-	Maximum Z axis reading
6...7	word	0...65535	-	Minimum Z axis reading

Message: SP-6 (Compass) Deviation compensation data packet 3				
Message ID: 4				**Data length:** 2 bytes
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	word	0...65536	-	Z axis current reading

In the case of the in-flight self calibrating firmware, only a single calibration packet is sent:

Message: SP-6 (Compass) calibration data packet 1	
Message ID: 2	**Data length:** 4 bytes

Byte	Type	Range / Value	Units	Description
0...1	word		-	Samples in calibration buffer
2...3	word	0...100%	-	Ellipsoid fit in percent

This packet is sent as long as calibration is active in addition to message ID 1. Samples in buffer refers to the number of unique three dimensional samples in the calibration data buffer. Samples are added based roughly on certain changes to the data based on previous samples collected.
The Ellipsoid fit gives an indication of the quality of samples in the buffer in order to describe the surface of a perfect ellipsoid. This value tends to increase as samples added or replaced with better ones and gives an indication of progress during the calibration process. Typical results after a suitable flight with a reasonable installation tend to be in the 96-98% region. 100% theoretically is unattainable due to sensor noise.

SP-7 Euler Attitude

The SP-7 will broadcast the Euler Attitude message 20 times per second. This is a default message format that all Stratomaster instruments expect. This message will be sent unconditionally.

Message: SP7 (AHRS) Broadcasts Euler Attitude (Absolute Attitude)				
Message ID: 3			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	int	-17999 ... +18000	1/100 deg	Euler Roll / Bank
2...3	int	-9000 ... +9000	1/100 deg	Euler Pitch
4...5	int	-17999 ... +18000	1/100 deg	Euler Yaw
6	byte	-50...50	-	Slip
7	bit 0	0 = accelerometer mode, 1 = gyro mode		
	bit 1	1 = SP-7 is in over range mode (gyro maximum rate exceeded)		
	bit 2	1 = SP-7 is at operating temperature (no longer used)		
	bit 3...4	Reserved		
	bit 5...7	000 = Unidentified AHRS, 001 = SP-7, Else reserved.		

An Euler Bank/Pitch/Yaw angle value of -9000 should be interpreted as -90.00 deg.

Gyro mode implies that the accelerometer corrections are not being applied. This mode is typically engaged when high rotation rates are being sensed. Accelerometer mode implies that the non-linear complimentary filters are being applied to the data (gravity vector is being used to correct/erect the horizon) – typically when the rotation rates are low. In both modes the measurements for the rate gyroscopes are being used to propagate the quaternion state vector.

RDAC XF messages

Note: If you are implementing a third party RDAC, please note that message timings are critical and must be exactly as shown (+/-10% variation is acceptable).
Messages ID 2-7 are collated at the iBOX into a single message that is sent over the iEFIS LAN. All of these messages must be created even if you are not using some of the contents. You must create all messages and send at the specified rates. It is recommended to send messages 2-7 as a single batch, one after the other (order does not matter) every 500 mS to reduce lag. Messages 2-7 contain data that does not change fast.

RDAC XF Message ID 1

Message: RDAC XF Message				
Message ID: 1			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	word	0-65536		Fuel flow 1 pulse count
2...3	word	0-65536		Fuel flow 1 pulse ratio
4...5	word	0-65536		Fuel flow 2 pulse count
6...8	word	0-65536		Fuel flow 2 pulse ratio

This message is sent every 4 seconds. Counts are totals over last 4 second period. Ratio is a number related to 1000 = 100.0%. For example: 400 = 60.0/40.0% ratio. No pulse detected = 65536.

RDAC XF Message ID 2

Message: RDAC XF Message				
Message ID: 2			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	sint		deg C	TC 1
2...3	sint		deg C	TC 2
4...5	sint		deg C	TC 3
6...8	sint		deg C	TC 4

This message is sent every 500mS.

Uncompensated thermocouple voltage scaled to degrees C based on K-Type probe.
Linear measurement. Probe voltage 24.9mV = 600 degrees C.
Add Temperature from message ID 7 after any conversion to other probe types to derive cold junction compensated temperature.

RDAC XF Message ID 3

Message: RDAC XF Message	
Message ID: 3	**Data length:** 8 bytes

Byte	Type	Range / Value	Units	Description
0..1	sint		deg C	TC 5
2...3	sint		deg C	TC 6
4...5	sint		deg C	TC 7
6...8	sint		deg C	TC 8

This message is sent every 500mS.

Uncompensated thermocouple voltage scaled to degrees C based on K-Type probe.
Linear measurement. Probe voltage 24.9mV = 600 degrees C.
Add Temperature from message ID 7 after any conversion to other probe types to derive cold junction compensated temperature.

RDAC XF Message ID 4

Message: RDAC XF Message				
Message ID: 4			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	sint		deg C	TC 9
2...3	sint		deg C	TC 10
4...5	sint		deg C	TC 11
6...8	sint		deg C	TC 12

This message is sent every 500mS.

Uncompensated thermocouple voltage scaled to degrees C based on K-Type probe.
Linear measurement. Probe voltage 24.9mV = 600 degrees C.
Add Temperature from message ID 7 after any conversion to other probe types to derive cold junction compensated temperature.

RDAC XF Message ID 5

Message: RDAC XF Message				
Message ID: 5			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	word	0-4095		OILT
2...3	word	0-4095		OILP
4...5	word	0-4095		AUX1
6...8	word	0-4095		AUX2

This message is sent every 500mS.

RAW ADC readings. Conversion to final value is up to equipment.

RDAC XF Message ID 6

Message: RDAC XF Message				
Message ID: 6			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	word	0-4095		FUELP
2...3	word	0-4095		COOLANT
4...5	word	0-4095		FUELLEVEL1
6...7	word	0-4095		FUELLEVEL2

This message is sent every 500mS.

RAW ADC readings. Conversion to final value is up to equipment.

RDAC XF Message ID 7

Message: RDAC XF Message				
Message ID: 7			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	sint		deg C	Temperature
2...3	word	See formula below		RDAC supply Voltage
4...8	word			Not used

This message is sent every 500mS.

//Pascal function to return voltage in 0.1V steps

```
function ToVolts(v: word): string;
begin
  result:=IntToStr(round(v/5.73758));
  if length(result)=1 then result:='0'+result;
  Insert('.',result,length(result));
end;
```

RDAC XF Message ID 8

Message: RDAC XF Message				
Message ID: 8			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	word			RPM1
2...3	word			RPM2
4...5	word	0-4095		MAP

6...8	word	0-4095		CURRENT

This message is sent every 200mS.

RPM1 and RPM2 is RPM based on 1 pulse per revolution.
if number is >=50000, value is scaled to RPM*10. Example: 51000 = 60000 RPM
Scaled 10 readings are normally used with turbine engines.

MAP and Current are RAW ADC readings. Conversion to final value is up to equipment.

RDAC XF Message ID 9
Note: This message exists in firmware from 29th October 2013 onwards

Message: RDAC XF Message				
Message ID: 9			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte			ID
1	byte			Spare
2...3	word			V1
4...5	word			V2
6...8	word			V3

Answer to request for calibration. Two IDs are used, 0 and 1.
ID 0
V1 – Ambient
V2 – TC
V3 – Analog
ID 1
V1 – MAP
V2 - VOLT

Flap and Trim controller

Setup 1 to Setup 7 messages must be requested by Host.
It is recommended for the host to request the setup from the SP10 rather than retain its own settings. It is possible for multiple control units (EFIS or other) to control a single SP10. Setups will be transmitted if they are changed. The setups tend to be required in order to draw current positions in graphic displays.
Status messages are sent at maximum interval of 1 second or if data has changed at minimum interval of 0.2 seconds.

Message: Setup 1				
Message ID: 8			**Data length:** 7 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0		Setup 1
1...2	byte	0-4095		FlapsUp

3...4	word	0-4095		FlapsMid1
5...6	word	0-4095		FlapsMid2
6...8	word	0-4095		FlapsDown

Message: Setup 2				
Message ID: 8			**Data length:** 3 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	1		Setup 2
1...2	word	0-4095		FlapsDown

Message: Setup 3				
Message ID: 8			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	2		Setup 3
1	byte	0-255		FlapsMotorTimeout
2	byte	0-255		FlapsMotorDir
3	byte	0-255		FlapsMotorForce

Message: Setup 4				
Message ID: 8			**Data length:** 7 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	3		Setup 4
1...2	word	0-4095		PitchUp
3...4	word	0-4095		PitchNeutral
5...6	word	0-4095		PitchDown

Message: Setup 5				
Message ID: 8			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	4		Setup 5
1	byte	0-255		PitchMotorTimeout
2	byte	0-255		PitchMotorDir
3	byte	0-255		PitchMotorForce

Message: Setup 6	
Message ID: 8	**Data length:** 7 bytes

Byte	Type	Range / Value	Units	Description
0	byte	5		Setup 6
1...2	word	0-4095		RollRight
3...4	word	0-4095		RollNeutral
5...6	word	0-4095		RollLeft

Message: Setup 7				
Message ID: 8			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	6		Setup 7
1	byte	0-255		RollMotorTimeout
2	byte	0-255		RollMotorDir
3	byte	0-255		RollMotorForce

Message: Flap Status				
Message ID: 0			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0		Function: Flaps
1	byte			Flags
2...3	word	0-4095		Position

Message: Pitch trim Status				
Message ID: 1			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	1		Function: Pitch trim
1	byte			Flags
2...3	word	0-4095		Position

Message: Roll trim Status				
Message ID: 2			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	2		Function: Roll trim
1	byte			Flags
2...3	word	0-4095		Position

Flag byte: Bit 0: 0=Motor off, 1=Motor on

Bits 1..3 - last commanded position + 1 from position command message (i.e. flapsUp, FlapsMid1,.... For all functions, Flaps, Pitch, Roll).

Message: Diagnostics (Raw ADC values)				
Message ID: 14 or 15			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..1	word	0-4095		ADC 0 or 4
2...3	word	0-4095		ADC 1 or 5
4...5	word	0-4095		ADC 2 or 6
6...7	word	0-4095		ADC 3 or 7

EFIS to Transponder messages

Note: If two transponders are fitted, these messages will be seen by both transponders, each transponder has its own reply HOST address.

Message: EFIS to Transponder Message (HOST address 1)				
Message ID: 5			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0..5	char	See below		Aircraft ID
6...7	sint	See below		Baro Altitude

This message must be send once per second. The transponder will be set to “OFF” or “Standby” state if this message is not received for 4 seconds.

Aircraft ID contains 8 consecutive characters coded by 6 bits each;
only capital letters A..Z, whitespace ' ' and numbers 0..9 are allowed
coding:
'A' = 0x01, ..., 'Z' = 0x1A, space = 0x20, '0' = 0x30, ..., '9' = 0x39
The eight resulting 6-bit numbers are concatenated into one big bitstring
and then split into six 8-bit chunks (bytes) for transmission
AID = aircraft ID, i.e. aircraft registration (e.g. N82381) or flight number
(e.g. UAL409), in the same format (without leading spaces, dashes, etc.) as
used on the filed flight plan
see also RTCA DO-181C paragraph 2.2.17.1.13c for coding
the resulting 7 byte string from RTCA DO-181C can be used, but without
the first byte (fixed 0x20). This gives 6 bytes finally

Altitude contains LSB first 16bit representation of 2's complement integer of
pressure altitude in 10ft steps. The negative value -101 must be used
for INVALID_ALTITUDE, due to missing or inaccurate altimeter data,
as determined by the altitude source

Message: EFIS to Transponder Message (HOST address 1)	
Message ID: 6	**Data length:** 8 bytes

Byte	Type	Range / Value	Units	Description
0..1	octal	0-7777		Binary squawk code
2	byte	See below		Category/size code
3	byte	0-4, Bits 3-7		Transponder state
4...6	hex	0-$FFFFFF		ICAO address 24 bits LSB first
7	byte	0-6, Bit 7		Aircraft Speed/ADSB in flag

This message must be send once per second. The transponder will be set to “OFF” or “Standby” state if this message is not received for 4 seconds.

Binary squawk code is transmitted as octal, three bits per digit (0..7) for a total of 4 digits = 12 bits. Byte order is LSB first. Octal 1234 is transmitted as $9C $02.

Aircraft catergory and size code:
1 byte, made up as follows:
Bits 0-2: L/W code as per DO260B, 0-15.
Bits 3-7: Aircraft category as per DO260B, 0-31.

Aircraft Speed:
0=unknown, 1= vmax<=75kt, 2= 75kt<vmax<=150kt, 3= 150kt<vmax<=300kt, 4=300kt<vmax<=600kt, 5=600kt<vmax<=1200kt, 6=vmax>1200kt

ADSB-IN flag:
BIT-7 of Aircraft Speed must be set if system has ADSB-IN traffic display capability.

Transponder state:
0: Off
1: Standby
2: Ground
3: ON
4: ALT

<u>Bits 3-7 are used as follows for messages ORIGINATING at the EFIS:</u>

3 – On 0-1 transition transponder will be set to TX IDENT for 18 seconds.
4 – If “0” transponder is set to “On ground” mode. “1” is “ in flight. Note: This bit MUST be set correctly for the relevant mode-s messages to be transmitted.
5 – 7 Not used.

<u>Bits 3-7 are used as follows for messages ORIGINATING at the transponder:</u>

3 – Set to 1 if a transponder reply occurred since the last ID 3 message sent, 0 otherwise
4 – Set to 1 if transponder failure (any failure reported by transponder itself)
5 – Set to 1 if transponder interface NOT receiving any data from transponder
6 – Set to 1 if IDENT is active
7 – if Mode S: If set to 1, at least 1 squitter was transmitted during last 0.5 seconds.
    If Mode C: Bit set to 0.

ALT may only be selected if altitude in message ID 1 is valid.

CAN messages for extended squitter (Mode-S transponders only)

Consists of three messages, must be sent once per second.

Message: EFIS to Transponder Message (HOST address 1)				
Message ID: 7			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	0		Message type
1	byte	0-3		GPS fix
2--3	word	0-3599	deg*10	Ground track
4--5	word	0-6553.5	Knots*10	Ground speed
6--7	word	0-65535	ft/5+1000	Height of geoid

Message: EFIS to Transponder Message (HOST address 1)				
Message ID: 7			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	1		Message type
1	byte	0-255		Number of satellites tracked
2	word	0-65535	ft/5+1000	Altitude HMSL
4--7	longint		MGL POS	Latitude

Message: EFIS to Transponder Message (HOST address 1)				
Message ID: 7			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	2		Message type
1--3	array	-		UTC hour:min:sec
4--7	longint		MGL POS	Longitude

MGL Position format: Integer 180000 = 1 degree of latitude or longitude. Negative is S or W, positive is N or E.
UTC format is binary hour:minute:second. Hour is first byte, second is last byte.
Heights are in steps of 5ft plus 1000. “0ft” is thus 1000. “1000 ft” is 200. “-100 ft” is 980.
Speed is in tenth of a knot.
Heading is in tenth of a degree.

These messages are not acknowledged (other than the normal CAN message ACK).

Messages originating at the transponder (HOST ID 44 or 45)

Message ID 5 is identical to Message HOST ID 5 but sent by the transponder (current state).
Message ID 6 is identical to Message HOST ID 6 but sent by the transponder (current state). Note: Exception bits 3...7 of Transponder state.

Messages sent by the transponder are sent in reply to messages received by the transponder. Should a message of either ID not be received by the transponder, the transponder will time out and send a unsolicited reply message every 4 seconds for the affected ID (which could include both IDs if not message is received).

ECB messages

Each of a possible 4 ECB units has 8 circuit breakers.
CAN addressing reserved is for 8 modules, 4 are used.

Message: ECB to EFIS Message (HOST address 1)				
Message ID: 0			**Data length:** 2 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte	One bit per breaker LSB=breaker 1		Breaker states (1=on/0=off)
1	byte	One bit per breaker LSB=breaker 1		Breaker tripped if 1
2	byte			Breaker doubling info
3,4	word		0.1V	ECB breaker input voltage

Message ID 0 is sent once every 4 seconds or whenever a breaker changes state.
Doubling info:
Bits 0 to 3, if a bit is set then the corresponding breakers are joined (applicable to breaker pairs 0,1 2,3 4,5 and 7,8).
If bit 4 is set then breakers 1,2,3 are a triple. If bit 5 is set then breakers 6,7,8 are a triple. If bit 6 is set then breakers 1,2,3,4 are a quad. If bit 7 is set then breakers 5,6,7,8 are a quad.
Breaker doubling, tripling, quading is set via DIP switch on the ECB module.

Message: ECB to EFIS Message (HOST address 1)				
Message ID: 1			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte		0.1A	Breaker 1 current in 0.1A steps
1	byte		0.1A	Breaker 2 current in 0.1A steps
1	byte		0.1A	Breaker 3 current in 0.1A steps
1	byte		0.1A	Breaker 4 current in 0.1A steps
1	byte		0.1A	Breaker 5 current in 0.1A steps
1	byte		0.1A	Breaker 6 current in 0.1A steps
1	byte		0.1A	Breaker 7 current in 0.1A steps
1	byte		0.1A	Breaker 8 current in 0.1A steps

Message ID 1 is send once per 4 second interval unless there is a change in one of the current readings in which case the message is sent at an interval no less than 0.5 seconds since the last message. In other words, this message is sent at intervals from 0.5 to 4 seconds depending on content changes.
Note: For doubled, tripled or quaded breakers, add the individual currents per breaker.

Message ID 2,3,4,5 is sent in response to a request for settings command

Message: ECB to EFIS Message (HOST address 1)				
Message ID: 2			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0 ... 1	word		0.1A	Breaker 1 current in 0.1A steps
2 ... 3	word		0.1A	Breaker 2 current in 0.1A steps
4 ... 5	word		0.1A	Breaker 3 current in 0.1A steps
6 ... 7	word		0.1A	Breaker 4 current in 0.1A steps

Message: ECB to EFIS Message (HOST address 1)				
Message ID: 3			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0 ... 1	word		0.1A	Breaker 5 current in 0.1A steps
2 ... 3	word		0.1A	Breaker 6 current in 0.1A steps
4 ... 5	word		0.1A	Breaker 7 current in 0.1A steps
6 ... 7	word		0.1A	Breaker 8 current in 0.1A steps

Message: ECB to EFIS Message (HOST address 1)				
Message ID: 4			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	byte		0.1S	Breaker 1 trip response time
1	byte		0.1S	Breaker 2 trip response time
1	byte		0.1S	Breaker 3 trip response time
1	byte		0.1S	Breaker 4 trip response time
1	byte		0.1S	Breaker 5 trip response time
1	byte		0.1S	Breaker 6 trip response time
1	byte		0.1S	Breaker 7 trip response time
1	byte		0.1S	Breaker 8 trip response time

Message: ECB to EFIS Message (HOST address 1)				
Message ID: 5			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**

0	byte			ECB Dipswitch reading
1	byte			WIGWAG breakers EVEN
2	byte			WIGWAG breakers ODD
3	byte		0.1S	WIGWAG time (0.1 Second steps)

Touchpad messages

Message: TouchPad to EFIS Message (HOST address 1)				
Message ID: 0			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	word			X
2...3	word			Y

Coordinates are send as whole numbers. Neither direction nor orientation are defined as this will depend on mounting. EFIS contains a user calibration to set this and limits as required.

Message: TouchPad to EFIS Message (HOST address 1)				
Message ID: 0			**Data length:** 4 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0	Gesture			

Gesture Types:

0x10 Single Tap

0x20 Double Tap

0x31 Swipe up

0x41 Swipe right

0x51 Swipe down

0x61 Swipe left

0x11 Tap and hold

0x32 Swipe up and hold

0x42 Swipe right and hold

0x52 Swipe down and hold

0x62 Swipe left and hold

iEFIS Extender messages

By default, the Extender uses its high speed data link to an EFIS and both CAN ports send data received from this link and pass any received data to this link. The CAN messages documented here are disabled. To enable them, send CAN message ID 10,11,12 or 13 (depending on CAN address set - “10” is the factory default for CAN address 64) and function ID 0 (“set outputs”) as documented in the EFIS to device section of this document. The Extender will then disable the high speed link and enable the CAN functionality documented here.

The Extender will send the value of its analog inputs in two messages, each containing 4 readings. These messages are sent every 125mS if at least one value (in a group of 4) has changed since the last transmission. If there is no change the next transmission will follow 1 second after the last.

Message: iEFIS Extender to EFIS Message (HOST address 1)				
Message ID: 0			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	word	0-4095		Analog input 1
2...3	word	0-4095		Analog input 2
4...5	word	0-4095		Analog input 3
6...7	word	0-4095		Analog input 4

Message: iEFIS Extender to EFIS Message (HOST address 1)				
Message ID: 1			**Data length:** 8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...1	word	0-4095		Analog input 5
2...3	word	0-4095		Analog input 6
4...5	word	0-4095		Analog input 7
6...7	word	0-4095		Analog input 8

The temperature is sent as degrees C added with 128. A value of 128 thus means 0 degrees. 127 is -1 degree, 129 is +1 degree.

Supply voltage is 5.0-30.5V in steps of 0.1V (byte value 0-255).

Pressure sensor is raw ADC reading from sensor.

Ports status uses bits 0-4 for RS232 ports 1-5. If bit is set then port has been initialized via setup message and is ready to operate.

Output states: Bits 0-4 reflect the status of the outputs 1-5. A “1” means the output is on. Output is an open collector output meaning “1” is output transistor is switched on and collector will be at ground level.

This message is sent at a rate of 125 mS unless there is no change to the data content in which case retransmission may be delayed by up to 1 second.

Message: iEFIS Extender to EFIS Message (HOST address 1)	
Message ID: 2	**Data length:** 6 bytes

Byte	Type	Range / Value	Units	Description
0...1	word	0-4095		Pressure sensor
2	byte			Supply voltage (see above)
3	byte			RS232 ports initialized status
4	byte			Output states
5	byte			Temperature (see above)

RS232 RX messages contain 1 to 8 bytes of received data on the relevant port.

Message: iEFIS Extender to EFIS Message (HOST address 1)				
Message ID: 3			**Data length:** 1-8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...7	bytes			RS232 RX port 1

Message: iEFIS Extender to EFIS Message (HOST address 1)				
Message ID: 4			**Data length:** 1-8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...7	bytes			RS232 RX port 2

Message: iEFIS Extender to EFIS Message (HOST address 1)				
Message ID: 5			**Data length:** 1-8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...7	bytes			RS232 RX port 3

Message: iEFIS Extender to EFIS Message (HOST address 1)				
Message ID: 6			**Data length:** 1-8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...7	bytes			RS232 RX port 4

Message: iEFIS Extender to EFIS Message (HOST address 1)				
Message ID: 7			**Data length:** 1-8 bytes	
Byte	**Type**	**Range / Value**	**Units**	**Description**
0...7	bytes			RS232 RX port 5

Note: RS232 ports must be initialized (baudrate etc) before they will operate. You do this using message ID 10 described in the EFIS to device messages section.

Appendix

Compass Tilt compensation

The SP-6/SP-7 combination can be used to implement a tilt compensated magnetic compass. The SP-7 bank and pitch angles are used with the three raw magnetic readings that have been compensated in the SP-6 for hard and soft iron distortions as well as offset and gain errors. This results in a fairly trivial solution for a system that can be used to derive correct magnetic heading during a banked turn and other aircraft manoeuvres.

The code presented here uses the bank and pitch angles from the SP-7 in a 0.1 degree resolution format and returns the heading in a 0.1 degree resolution.
The code makes no use of floating point, only 32 bit integer arithmetic is required to suit typical microcontrollers.

Missing from this code are SIN and COS routines which are left to the reader to implement. Routines should accept angles in 0.1 degree resolution and return a result from -65535 to +65535 (I.e. scaled as 16:16 fixed point). This can be done easily by means of a precomputed lookup table.

```
var
  WMagARG,RawHeading: longint;
  EW,NS: longint;


//Fixed point 16:16 multiplication
function Mult(a,b: longint): longint;
begin
   result:=(int64(a)*int64(b)) div 65536;
end;
const
  ArcTanTable: array[0..46] of word = (
    0,17,35,52,70,87,105,123,141,158,
    176,194,213,231,249,268,287,306,325,344,
    364,384,404,424,445,466,488,510,532,554,
    577,601,625,649,675,700,727,754,781,810,
    839,869,900,933,966,1000,1036);

//Integer based fast ARCTAN2 function

function FArctan2(V1,V2: longint): longint;
var
  i,Quadrant: byte;
  neg: boolean;
  WTempLong,WMarg1,WMarg2,Ratio: longint;
begin
  Quadrant:=0;
  neg:=false;
  if V1<0 then
  begin
    Quadrant:=2;
```

```
    V1:=abs(V1);
  end;
  if V2<0 then
  begin
    Quadrant:=Quadrant or 1;
    V2:=abs(V2);
  end;
  if V1>V2 then
  begin
    Ratio:=V1;
    V1:=V2;
    V2:=Ratio;
    Neg:=true;
  end;
  Ratio:=V1*1000 div V2;
  i:=0;
  WTempLong:=ArcTanTable[0];
  while (Ratio>WTempLong) and (i<45) do
  begin
    inc(i);
    WTempLong:=ArcTanTable[i];
  end;
  WMarg1:=ArcTanTable[i+1]-WTempLong;
  WMarg2:=WTempLong-Ratio;
  if WMarg2>0 then
  begin
    WMarg2:=WMarg2 shl 16; //mult by 65536
    WMarg2:=WMarg2 div WMarg1;
  end;
  WMarg1:=i shl 16;
  WMarg2:=WMarg1-WMarg2;
  if neg then WMarg2:=5898240-WMarg2;
  case quadrant of
  0: result:=WMarg2;
  1: result:=11796480-WMarg2;
  2: result:=-WMarg2;
  3: result:=-(11796480-WMarg2);
  end;
end;

//returns heading in variable “Angle”
//Note that on entry “Angle” is set to the SP-6 accelerometer based tilt compensated
//heading divided by 10 (I.e. in 0.1 degree resolution.

procedure TiltCompensateCompass(var Angle: longint);
var
  MagPitch,MagBank,WTempLong,Heading: longint;
begin
  MagPitch:=RawPitchAngle; //pitch and bank from AHRS in 0.1 degrees
  MagBank:=-RawBankAngle; //note sign reversal
  WMagARG:=icos(MagPitch);
  NS:=Mult(MagRawX,WMagARG);
```

```
  WMagARG:=isin(MagBank);
  WTempLong:=Mult(MagRawY,WMagArg);
  WMagARG:=isin(MagPitch);
  NS:=NS+Mult(WTempLong,WMagArg);
  WMagARG:=icos(MagBank);
  WTempLong:=Mult(MagRawZ,WMagArg);
  WMagARG:=isin(MagPitch);
  NS:=NS-Mult(WTempLong,WMagArg);

  WMagARG:=icos(MagBank);
  EW:=Mult(MagRawY,WMagARG);
  WMagARG:=isin(MagBank);
  EW:=EW+Mult(MagRawZ,WMagARG);
  Heading:=(Farctan2(EW,-NS));
  if Heading<0 then Heading:=Heading+23592960;
  if Heading>=23592960 then Heading:=Heading-23592960;
  Heading:=Heading div 6554;

 //We now have heading derived from mag sensor X/Y/Z
```

{The following code averages compass accelerometer based tilt compensation with that derived from the above calculation for angles up to 15 degrees of bank with the ratios dependent on the angle. This allows a good changeover between non-accelerated (but very accurate) and accelerated flight regime. The premiss here is that often the AHRS and compass are not mounted on the exact same plane and this results in better accuracy of the accelerometer based solution (done inside the compass) in straight and level flight. This is considered optional.)

```
  if abs(MagBank)<=150 then
  begin
    if (heading<900) and (Angle>2700) then
    begin
      Angle:=Angle-3600;
    end else
    if (heading>2700) and (Angle<900) then
    begin
      Heading:=Heading-3600;
    end;
    f1:=abs(MagBank);
    f2:=150-f1;
    heading:=(Heading*f1) + (Angle * f2);
    Heading:=heading div 150;
    if heading>3599 then heading:=heading-3600 else
   if heading<0 then heading:=heading+3600;
  end;

 //return the result
  Angle:=Heading;
end;
```