MGL Autopilot servo interface protocols

This document provides interface details allowing the use of MGL Avionics servos for third party applications.

There are conditions attached to using the protocol as well as the MGL servos.

1) You accept any and all responsibility for direct and indirect use of your application and hold free of all responsibility MGL Avionics and its representatives for any cause including negligence by any party.
2) You will perform practical tests to ascertain the suitability of both the MGL servo as well as your application for its intended use including performing wear tests to measure the expected lifetime of the system in your application.
3) You will inform any of your application users of any limitations found and provide a schedule of maintenance checks on the servo mechanics in order to ensure safe operation of your system at all times.
4) You will consult MGL Avionics on your intended use if your application would use the servo in an unusual way that may impact its performance or durability.
5) You will never use the servo in an application where its failure, howsoever caused, may result in damage to property, injury or death.
6) You will ensure that any form of servo failure will not cause harm cited under (5) by means of external safety mechanisms such as shear pins, clutches, couplings with designed failure points etc.
7) You will not use or encourage use of your application or MGL servo for any military use including indirect use such as training. If you are not certain if your application falls under this restriction (for example law enforcement) please contact MGL Avionics for a resolution.
8) You are in control of the servos use. You accept all responsibility. If you do not agree with this, you do not have permission to use the MGL servo.

Servo RS232 communications protocol

Up to 4 servos can be connected to one RS232 port. The Servo responds to 2 different message types in order to control identity / function, position, torque setting, engage / disengage, etc.

The data structures of the RS232 servo communications protocol is based on the servo CAN protocol, but not exactly equivalent.

Baud rate is 38400 baud. Transmission format is asynchronous, 1 start bit, 8 data bits, 1 stop bit. No Parity. The output circuit is given in figure 1 (below):

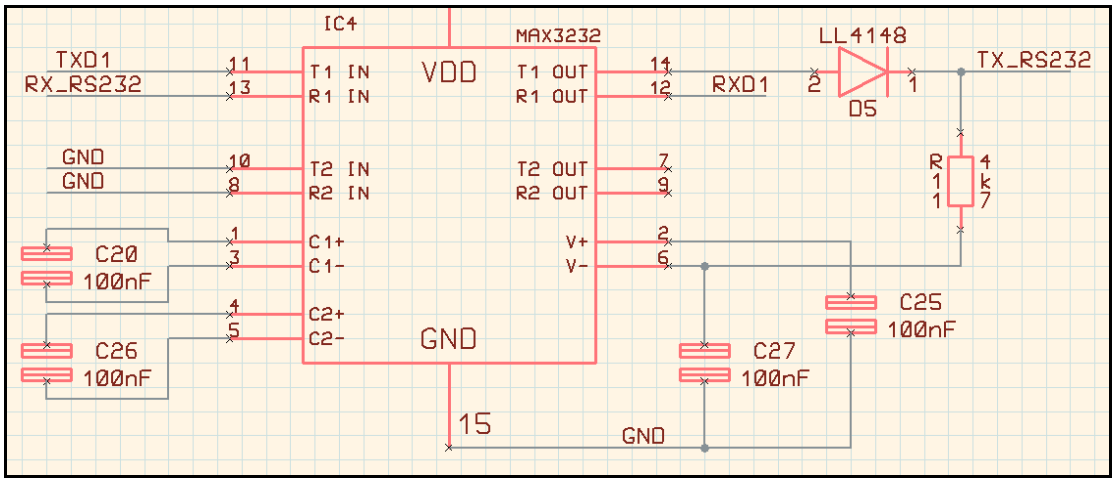

Figure 1: Servo RS232 Output Stage

Data formats used in message protocol:

CKS1: Checksum from Message type to last data byte before CKS1. Seeded with $AA. Addition of all bytes in range modulo 256.

CKS2: Checksum from Message type to last data byte before CKS1. Seeded with $55. Logical XOR of all bytes in range.

Message: Set Servo Number (Host/EFIS to Servo)

$D5	DLE
$82	STX (Transmission Start)
$06	Data Length
$00	Message Type
$00	Sender Address: Host / EFIS is Address 0
$AA	$AA
$55	$55
SRVNUM	Byte - Servo Number (0 - 16)
NSRVNUM	Byte - SRVNUM XOR $FF
CKS1	Byte - linear checksum
CKS2	Byte - XOR checksum

Message: Send 4 Servo Positions (Host/EFIS to 4x Servos)

$D5	DLE
$82	STX (Transmission Start)
$0F	Data Length (15)
$01	Message Type
$00	Sender Address: Host / EFIS is Address 0
FRESPOND	bits 0 -> 3 used as flags indicating which of the 4 servos should respond: 1 => respond, 0=> don't respond bit 0: servo 1, bit 1: servo 2, bit 2: servo 3, bit 3: servo 4
OPTIONS1	Servo 1... Bit 0: 0 => do not engaged, 1 => engage Bit 1: 0 => no action, 1 => reset Torque measurement Bit 4 - 7: Torque
TGT1LSB	Byte - Target Position Servo 1 Position (LSB)
TGT1MSB	Byte - Target Position Servo 1 Position (MSB)
OPTIONS2	Servo 2... Bit 0: 0 => do not engaged, 1 => engage Bit 1: 0 => no action, 1 => reset Torque measurement Bit 4 - 7: Torque
TGT2LSB	Byte - Target Position Servo 2 Position (LSB)
TGT2MSB	Byte - Target Position Servo 2 Position (MSB)
OPTIONS3	Servo 3... Bit 0: 0 => do not engaged, 1 => engage Bit 1: 0 => no action, 1 => reset Torque measurement Bit 4 - 7: Torque
TGT3LSB	Byte - Target Position Servo 3 Position (LSB)
TGT3MSB	Byte - Target Position Servo 3 Position (MSB)
OPTIONS4	Servo 4... Bit 0: 0 => do not engaged, 1 => engage Bit 1: 0 => no action, 1 => reset Torque measurement Bit 4 - 7: Torque
TGT4LSB	Byte - Target Position Servo 4 Position (LSB)
TGT4MSB	Byte - Target Position Servo 4 Position (MSB)
CKS1	Byte - linear checksum
CKS2	Byte - XOR checksum

Message: Acknowledge Servo Position Command (Servo to Host / EFIS)

$D5	DLE
$82	STX (Transmission Start)
$07	Data Length
$01	Message Type
SRVNUM	Byte - Sender Address: Servo Number (1 - 16)
STATUS	Bit 0: 0 => not engaged, 1 => engaged Bit 1: 0 => not slipping, 1 => slipping Bit 2: 0 => Voltage OK, 1 => Voltage Alarm
POSLSB	Byte - Current position LSB
POSMSB	Byte - Current position MSB
VOLT	Byte - Voltage in steps of 0.1V + 5 (5.0 - 35.5V = 0 - 255)
TORQ	Byte - Measured Torque (-60 to 60)
CKS1	Byte - linear checksum
CKS2	Byte - XOR checksum

In order order to prevent data collision, servo 1 (bank) will respond immediately after the position command has been received. The pitch / number 2 servo will respond 10ms after receiving the "4 servo" command. Servo #3 (yaw) will respond 20ms and servo #4 (throttle/other) 30ms after the position command is received.

Notes:

Positions are in the range of 0-4095. Positions are arbitrary and not related to a particular position of the shaft (but positions are constant for a given servo). Different servos can have different positions for identical shaft positions.

Servo numbers range from 0 to 16. Number 0 is only used by the servo identification message to set a servo to unidentified, virgin state.

For RS232 applications a maximum of 4 servos can be used which implies that only servo numbers 1 to 4 are sensible. Numbers 5 to 16 can only be used in CAN bus applications.

MGL Servo CAN bus protocol

MGL servos operate using standard CAN bus protocol at 250KBaud. Please note that servos do not 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.

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.

The Host logical device shall be device address 1. All servos shall receive this message.
Servos will occupy addresses 16-31. The address is the (servo number – 1 +16).

Messages Host -> servo

Message ID 0 -> set servo number

Data length: 4 bytes.
Byte 0 - 0xAA
Byte 1 - 0x55
Byte 2 - Servo number 0 – 16
Byte 3 - Byte 2 XOR 0xFF

Notes: A new servo will have number 0 and will not respond to any position messages.

Reception of this message will set the servo number in the servos non-volatile memory.
If the servo is set to the requested number already then the servo will not write to non-volatile memory.
The servo shall accept servo number 0 in order to set a servo into “virgin” state.

The host can identify the success of this operation as it will start receiving position reply messages in response to outgoing position messages.
The recommended sequence would be:

1) Send Message ID 0 containing desired servo number
2) Wait at least 100mS
3) Send Position message (ID 1)
4) Receive Position message from servo (ID 1)

Message ID 1 -> set position

Data length: 4 bytes.
Byte 0 - Servo number 1-16
Byte 1 - Options byte
Bit 0: 0: Do not engage. Bit 1: Engage servo.
Bit 1: 1: Reset torque measurement
Bit 4-7: Torque
Byte 2 - LSB position
Byte 3 - MSB position

Notes: The servo shall respond with the current position message if it is addressed by the servo number.

Messages Servo – Host

The servo address range is from 16 to 31 ((servo number-1)+16). Servos that have an address of 0 will never respond to any message until the servo address is set.

Message ID 1 -> Set position reply message

Data length: 5 bytes.
Byte 0: Status byte
Bit 0: 0=not engaged, 1=engaged
Bit 1: 0=not slipping, 1=slipping
Bit 2: 0=Voltage OK, 1=Voltage alarm
Byte 1: Current position LSB
Byte 2: Current position MSB
Byte 3: Voltage in steps of 0.1V+5V 5.0-35.5V = 0-255
Byte 4: Measured torque

Servo identification

A newly manufactured servo or a servo that has been reset to virgin state will not respond to any position message until it has been “identified”.

The process involves connecting a single servo to the communications bus (CAN or RS232) and then sending the identification message containing the desired servo number (1-4 for RS232, 1-16 for CAN).

Once a servo has been identified, it retains this identification for future use until changed by a new identification message.

Once all servos in a system have been identified, they can be joined onto the common communications bus.

In MGL EFIS systems, servo 1 is used as bank servo while servo 2 is used as pitch servo.

Notes:

The LED on the servo will show steady if no valid position request has been received and it will flash if a position request has been received.

A virgin servo (with servo number 0) will not receive any positioning message and so the LED will remain steady on.

For typical autopilot applications it is recommended to send a position request to the servo 10 times per second even if position has not changed as the servo will disengage if it does not receive regular position requests.

The latest position request takes priority, even if a previously requested position has not yet been reached.

The servo controlling host system must be able to handle position wraps from 4095 to 0 and 0 to 4095 correctly. Positioning will be performed in a direction that will result in the fewest steps required.

For positioning in excess of 180 degrees, the host must first position in the desired direction with a position offset of less than 2048, wait for this position to be reached (or close to this position) before issuing a further position.

There is no limit to the number of turns permitted.

Establishing servo wear limits

In order to provide a safe application, it is required to establish how long it takes until a servo will fail due to wear or other effects.

It is understood that this depends heavily on the environment that the servo is used in. For example, exposure to a corrosive atmosphere such as found close to the ocean may have an impact.

Accelerated wear testing is acceptable provided temperatures are not allowed to rise above normal operating temperature of the parts.

Wear points:

1) Contact points between steel gear on stepper motor and nylon gear on output shaft. Inspect the nylon gear for wear of the teeth. Wear limit is a 50% reduction in the width of the teeth from new. It is acceptable to rotate the output to use a fresh, unused part of the gear for applications that use only a fraction of a rotation (maintenance check and item).
2) Output shaft bearings. Two sealed high quality ball bearings are used on the output shaft. Check for ball or race breakage (rough turning of the shaft). This is not normally a wear issue but operation in a corrosive atmosphere may impede life expectancy.
3) Stepper motor. Check for free turning. There are no contacts or brushes to wear.
4) Output shaft position as well as force feedback measurement are performed using contactless methods so there is no wear on these items.
5) Check for effects of corrosion on the electronic PCB if exposed to a corrosive environment.

Perform a test using typical expected loads (bucket of water lifted and lowered with the servo on its side is a good and simple way). Run the servo for a few days lifting and lowering the load at a rate that would approximate typical expected servo movement, periodically check the servo and servo gear for signs of wear until you can positively measure the wear. At this point you can estimate the expected durability of the servo in your application.

Do not run the motor continuously in the accelerated wear test as the friction between gears will cause the nylon gear to heat up which will reduce the life time to less than what you would expect in a real life application. Move the motor, pause briefly, move it again, pause and so on. This will allow heat to dissipate and approximates a real application servo movement.