Atmel Servo motor controller V0.0 (Draft)

Name	Date	Version	Changes
Bill Arden	Dec-27-2002	0.0	Initial draft

1 Overview

The Atmel servo is designed to take commands from either a serial interface or a R/C input and use it to control a DC motor using a R/C type motor controller. A quadrature encoder is used to precisely control the speed and position.

This system has numerous applications like two-wheeled Battlebots using R/C control and general Robotic applications using the serial command interface.
It can even be controlled by both the R/C interface and the serial interface at the same time, which makes it possible to use the R/C radio as a manual/ override input.

The Atmel servo is designed to be compatible with the http://www.jrkerr.com/ devices. The serial protocol is compatible and the command interface has only slight differences. This means that you can mix JrKerr devices and this Atmel servo on the same buss if you have the proper interfacing adapters.

1.1 Serial Interface

The Atmel servo uses a 5-volt full duplex serial buss. An adapter converts the RS232 from the PC to the 5-volt buss. The Atmel servo tristates it's transmit pin when not sending a response.

A standard daisy chain technique is used to set the device addresses. This is similar to the JrKerr system except that the address in pin is grounded at the host so that the first device on the buss has and Address of "1" and that there is no need to ground the pin on the last device.

1.2 PID

2 Hardware

2.1 Atmel Servo

The Atmel servo is designed to have very few external components to make prototyping easier. The firmware is designed to run on an Atmel AT90S2313 running at 9.216Mhz.

See the schematic for more details

3 Interfaces

3.1 R/C input

A standard 3 pin connector is provided to connect to a R/C receiver.
The signal is the standard .5 ms to 1.5ms pulse that servos use.

3.2 Encoder Input

The Quadrature encoder input is interrupt driven and can count at over 500K transitions/second or 125K lines per second.

3.3 R/C output

The R/C output produces a standard .5 ms to 1.5mS pulse that R/C controllers like the Victor 888 needs. The output can drive 20mA and can run the victor directly

3.4 PWM output (optional)

The PWM output creates a pulse width modulated output with a direction pin that can drive H bridge controllers.

Note: R/C input and output are not available when PWM mode is used.

3.5 Serial Interface

The Atmel servo is controlled using packets.
Each packet has an “To” address , a command, a length, and a checksum at the end

Words are sent LSO (least significant octet) first.

3.5.1 Command Packet definition

Start Code	0xAA to start the packet
Address	1 to 127 for normal addresses 128 to 254 for group addresses 255 for broadcast to all
Command/Len	Lower nibble is the command code. (see below) Upper nibble is the Length. 0 = zero data bytes
Data*n	0 to 15 data bytes depending on the command
Checksum byte	Sum of Address to end of data. (Start code is excluded)

3.5.2 Status Packet definition

Status Byte	
Pos 4 bytes	Current Position (If enabled)
A/D 1 byte	A/D (R/C input) value (If enabled)
Vel 2 bytes	Current velocity (If enabled)
Aux 1 byte	Auxiliary status byte (If enabled)
Home 4 bytes	Home position (If enabled)
Type 2 bytes	Device type (If enabled)
Checksum byte	Sum of status byte and any extra bytes

Note: Status packets are not returned for a group command unless there is a group leader.

3.5.3 Addressing

At power up or after a “Hard reset command” all devices are at a address of “0” and only the first device will respond since its Select input is grounded.
So use a “set address” command addressing device 0 and set it to address “1”
Once the devices address is set it will select the next device so set the next device to “2”
Repeat until there are no more devices.

3.5.4 Baud rate

The baud rate defaults to 19200.
Send a broadcast “Set baud rate” command to set all the controllers to a new baud rate.

4 Serial interface Commands

(hex)	Command	Returns (unless sent as a group command)
00	Reset position	Status
01	Set Address [Address][group]	Status
02	Set Status Type [x]	Status
03	Get Status [x]	Custom Status
04	Load trajectory [Control][0 to 13]	Status
05	Start Motion	Status
06	Set Gain [14]	Status
07	Stop motor [control] [0 or 4]	Status
08	I/O control [1] **(not used)**	Status
09	Set Home mode [1] **(not used)**	Status
0A	Set Baud rate [Baud byte]	Group command only. No reply
0B	Clear Status Sticky bits	Status
0C	Save current position as home	Status
0D	Debug memory read/ write	Debug info
0E	Return last status	Status
0F	Reset	None

Address
Atmel will start at an address of 00 (hex)
The Select in and select out pins are daisy chained from the controller.
The controller grounds the select in of the first unit.

4.1 Reset Position (0x00)

This command set's the encoder position and target position to "0"
If issued during a trapezoidal profile motion the motor will stop.

4.2 Set Address(0x01)

Data byte 1 = Address 1 to 127
Data Byte 2 = Group address 0x80 to 0xFF. Clear top bit if group leader.

If the address is "0" then the module will tristate Select in and select out.

4.3 Define Status (0x02)

This command sets what items are returned with status.

Data Byte 1 = Status items. (Add or "or" the hex numbers together)

Bit	(Hex)	Size	True to send
Bit 0	0x01	Long 4	Current encoder Position, LSO
Bit 1	0x02	Byte 1	Send A/D value. (R/C input)
Bit 2	0x04	Short 2	Current Velocity. (Counts per servo tick)
Bit 3	0x08	Byte 1	Auxiliary status byte
Bit 4	0x10	Long 4	Saved encoder Position
Bit 5	0x20	Word 2	Send Device ID (Atmel Servo = 0)
Bit 6			
Bit 7			

The reply status will reflect the changes.
At power up the status items will default to "0"
See the status packet definition for more on the status information.

4.4 Read Status (0x03)

Similar to "Define status" except that the status list will not be saved.
This command returns a one-time status message.

See "Define status" for more information.

Data Byte 1 = Status items. (Add or "or" the hex numbers together)

Bit	(Hex)	Size	True to send
Bit 0	0x01	Long 4	Current encoder Position, LSO
Bit 1	0x02	Byte 1	Send A/D value. (R/C input)
Bit 2	0x04	Short 2	Current Velocity. (Counts per servo tick)
Bit 3	0x08	Byte 1	Auxiliary status byte
Bit 4	0x10	Long 4	Saved encoder Position
Bit 5	0x20	Word 2	Send Device ID (Atmel Servo = 0)
Bit 6			
Bit 7			

4.5 Load trajectory (0x04)

This command performs various types of moves.

As bits are set in the control byte additional data bytes will need to be sent in the order listed.

Data Byte 1 = Control byte. (Add or "or" the hex numbers together)

Bit	(Hex)	More data	
0	0x01	4 bytes	Load target position for a trapezoidal profile move.
1	0x02	Unsigned 4 bytes	Load target velocity for a trapezoidal profile move.
2	0x04	Unsigned 4 bytes	Load acceleration for a trapezoidal or velocity move.
3	0x08	Unsigned 1 Byte	Load PWM (R/C out magnitude) value. PWM output is set to "0" if Position servo is disabled
4	0x10		Servo mode 0 = manual mode. (See bit 3) 1 = Position servo mode
5	0x20		0 = trapezoidal move 1 = velocity move
6	0x40		0=PWM or R/C out FWD 1= PWM or R/C out REV
7	0x80		1 = start move now

4.6 Start motion (0x05)

This command is the same as setting bit 7 in the "Load trajectory" command.
This command is useful for loading multiple trajectories and then starting the motors at the same time with a group command.

4.7 Set Gain (0x06)

This command set's parameters for the PID and limit settings.

Bytes	
1,2	"Kp" PID Position gain.
3,4	"Kd" PID Velocity gain.
5,6	"Ki" PID Integral gain.
7,8	"IL" PID Integration limit. (This is multiplied by 256 before being used)
9	"OL" Output limit.
10	"CL" Current limit. (not used)
11,12	"EL" Position Error limit.
13	"SR" Servo rate divisor. (1.953Khz / x) = rate in Hz
14	Amplifier dead band compensation. Positive to jump between –x and +x Negative to create a dead band.

4.8 Stop Motor (0x07)

Stops the motor

Data Byte 1 = Control

Bit	Hex	More data	
0	0x01		Amplifier enable 0 = sets "AMP_EN" to low 1 = sets "AMP_EN" to high
1	0x02		Turn off position servo 1 = Disable position servo and set PWM to "0".
2	0x04		Stop abruptly 1 = set current command velocity and target velocity to "0" Enables position servo and enters velocity mode.
3	0x08		1= Stop smoothly by setting target velocity to "0" Enables position servo and enters velocity mode.
4	0x10	4 bytes	1= Jump to x (Causes motor to jump to new position.) Enables position servo and enters velocity mode. Note: this can be used to create crude motions.
5	0x20		
6	0x40		
7	0x80		

4.9 I/O control (0x08) (not implemented)

This command controls the configuration of limit1 and limit2 to be either inputs or outputs.

Data Byte 1 = Control

Bit	Hex	Set to
0	0x01	Output value of limit1
1	0x02	Output value of limit2
2	0x04	Direction of limit1 pin 0=output 1= input
3	0x08	Direction of limit2 pin 0=output 1= input

4.10 Set Home mode (0x09) (not implemented)

Sets how the home point is controlled.

Data Byte 1 = Control

Bit	Hex	Set to
0	0x01	Capture home on change of limit1 input
1	0x02	Capture home on change of limit2 input
2	0x04	Turn motor off at home. “AMP_EN”
3	0x08	Capture home on change of encoder index input.
4	0x10	Stop abruptly on home position.
5	0x20	Stop Smoothly on home position.
6	0x40	Capture home when an excess position error occurs
7	0x80	

4.11 Set Baud Rate (0x0A)

This command sets the baud rate.
This is usually sent as a broadcast since any reply would be at the new baud rate.

Data Byte 1 = baud rate devisor

Baud rate	Number	Notes
9600	129	
19200	63	Default
38400	30	
57600	20	
115200	10	

4.12 Clear status sticky bits (0x0B)

This command clears the bits in the status byte that are latched such as “over current”, “Position Error”, “Position overflow”, “Servo timer overrun”. These bits will stay set until this command is sent.

4.13 Save current position as home (0x0C)

Saves current encoder position as “Home”
This is useful to synchronize reading of the encoder position by sending this as a group command.

4.14 Debug command (0x0D)

This command reads or writes to data ram for diagnostic purposes.
To read
Data Byte 1,2 = Address to read from.
Data Byte 3 = length
To Write
Data Byte 1,2 = Address to write to + 0x8000
Data*n = data to write. (packet length is used to determine length)

4.15 Send Status (0x0E)

This command returns a status packet.

4.16 Hard reset (0x0F)

This command sets unit back to the power up state.

5 Appendix A

5.1 Status byte definitions

Bit	Hex	Name	Definition
0	0x01	move_done	Clear if a trapezoidal move is in progress or if accelerating or decelerating in velocity mode Set if motor is not moving or if PID is disabled.
1	0x02	cksum_error	Set if there was a checksum error in the previous command packet. (If the checksum is wrong there is no reply)
2	0x04	overcurrent	(not used) Set to “0” Must be cleared using the “clear sticky bits" command.
3	0x08	power_on	(Not used) Set to “1”
4	0x10	pos_error	Set if the position error exceeds the position error limit. Set if PID is disabled. Must be cleared using the “clear sticky bits" command.
5	0x20	Limit1	N/A Set to “0”
6	0x40	Limit2	N/A Set to “0”
7	0x80	Home in progress	N/A Set to “0”

Notes:

5.2 auxiliary status definitions

Bit	Hex	Name	Definition
0	0x01	Index.	Inverted value of the encoder index input.
1	0x02	pos_wrap	Set if the 32 bit position counter wraps or overflows. Must be cleared using the “clear sticky bits" command.
2	0x04	servo_on	“1” if the position servo is enabled.
3	0x08	accel_done	Set when the initial acceleration phase of a trapezoidal profile move is complete. Cleared when the next move is started.
4	0x10	slew_done	Set to “1” when the slew? portion of a trapezoidal profile move is complete. Cleared when the next move is started.
5	0x20	servo_overrun	Set to “1” when processor runs out of processing time. Cleared when the next move is started.
6	0x40		Set to “0”
7	0x80		Set to “0”

Notes