Brief:

The new firmware for AX12 servo motor is created with the goal of improvement of the speed and the flux of communication between Servos and Master.

We also pretend to have a different approach of control, not just a static control but also dynamic control, meaning this not just a control of a position or the compliance behavior when on the goal position but also the speed when dislocating to the goal position.

**Communication protocol packet:**

The communication protocol packet has been idealized to allow multi-master communication, and has the following composition in bytes:

[Header] [Packet Length] [Destination ID][Origin ID][Command][Data 1]…[Data N] [CRC]

Length

CRC

[Header] is always 0xAA

[Packet Length] is the Length of the whole packet less the header, the length itself and the CRC, in bytes.

[Destination ID] is the ID of the packet destination

[Origin ID] is the ID of the device who sent the packet

[Command] is the code of query or reply command

[Data] is/are the bytes of data, if any

[CRC] is the CRC of the whole packet less the header and the CRC itself.

The CRC is the sum of every negated bytes (of the whole packet, less the header and itself).

Ex:

Packet 0xAA 0x03 0x01 0x0FE 0x01 0xF9

0xAA is the header

0x03 The Length

0x01 The Destination ID

0xFE Master ID

0x01 PING request

0xF9 Result from ~(0x03) + ~(0x01) + ~(0xFE) + ~(0x01) = 0xF9

The servo code performs on this first version, reply to just a few commands.

**Request and Reply Commands and codes:**

PING 0x01

READ 0x02

WRITE 0x03

START 0x06

CYCLE 0x07

OK 0x04

NOK 0x08

RESET 0x09

SYNC\_WR 0x83

**PING**

Ping command has to be sent by regular ID not broadcast, it’s a simple why to check the presence of a device on the network.

A reply to this command will be the PING command itself, or in case of erroneous CRC, a NOK.

Ex:

Master to Servo

Packet 0xAA 0x03 0x01 0x0FE 0x01 0xF9

0xAA is the header

0x03 The Length

0x01 The Destination ID

0xFE Master ID

0x01 PING request

0xF9 Result from ~(0x03) + ~(0x01) + ~(0xFE) + ~(0x01) = 0xF9

Servo to Master

Packet 0xAA 0x03 0xFE 0x01 0x01 0xF9

0xAA is the header

0x03 The Length

0xFE The Destination ID

0x01 Master ID

0x04 OK reply

0xF6 Result from ~(0x03) + ~(0x01) + ~(0xFE) + ~(0x04) = 0xF6

**READ**

Read command has to be sent with regular ID, and allow to read data from the device (both eeprom and ram), given a memory address and a length of bytes to read, never being the sum of the address plus the length of data to read, bigger than the maximum address possible.

A reply to this command will be the data requested, or a NOK in the case of erroneous CRC or in case of the sum of the address plus the length of data to read being bigger than the maximum address possible.

Ex:

Master to Servo

Packet 0xAA 0x05 0x01 0x0FE 0x02 0x0B 0x02 0xZZ

0xAA is the header

0x05 The Length

0x01 The Destination ID

0xFE Master ID

0x02 READ request

0x0B Memory Address to read (Position on this example)

0x02 Length of data to read in bytes

0xZZ CRC

Servo to Master

Packet 0xAA 0xNN 0xFE 0x01 0xn1 0xn2 … 0xnn 0xZZ

0xAA is the header

0xNN The Length

0xFE The Destination ID

0x01 Slave ID

0xn1 data 1

0xn2 data 1

…

0xnn data n

0xZZ CRC

**WRITE**

Write command can be sent either by regular ID or broadcast, and allow to write data to the device (both eeprom and ram), given a memory address and a length of bytes to write, never being the sum of the address plus the length of data to read, bigger than the maximum address possible.

A reply to this command will an OK, or a NOK in the case of erroneous CRC or in case of the sum of the address plus the length of data to read being bigger than the maximum address possible or even if any parameter to be written is out of the correspondent limits.

NOTE:

The reply from the SERVO will already be done with the new settings.

Master to Servo

Packet 0xAA 0xXX 0x01 0x0FE 0x03 0x0B 0xn1 0xn2 … 0xnn 0xZZ

0xAA is the header

0xXX The Length

0x01 The Destination ID

0xFE Master ID

0x03 WRITE request

0x0B Memory Address to start writing

0xn1 data 1

0xn2 data 2

…

0xnn data n

0xZZ CRC

Servo to Master

Packet 0xAA 0x03 0xFE 0x01 0x04/0x08 0xZZ

0xAA is the header

0x03 The Length

0xFE The Destination ID

0x01 Slave ID

0x04 OK reply

Or

0x08 NOK reply If any of the values is located out of the memory range

0xZZ CRC

**START**

Start command has to be sent by broadcast, and is used to init the bus, once after this every servo will send a PING to the START requester, with a delay proportional to the ID, allowing like this to the MASTER build a table of existing devices, and to each SERVO their turn on the bus.

A reply to this command will be a PING from each connected SERVO within the ID correspondent delay.

Master to Servo

Packet 0xAA 0x03 0x01 0x0FE 0x06 0xZZ

0xAA is the header

0x03 The Length

0x01 The Destination ID

0xFE Master ID

0x06 START request

0xZZ CRC

Servo1 to Master

Packet 0xAA 0x03 0xFE 0x01 0x01 0xF9

0xAA is the header

0x03 The Length

0xFE The Destination ID

0x01 Slave ID

0x01 PING reply

0xF6

Servo 2 to Master

Packet 0xAA 0x03 0xFE 0x02 0x01 0xZZ

…

Servo N to Master

Packet 0xAA 0x03 0xFE 0xnn 0x01 0xZZ

NOTE: All servos will reply separated by the maximum of 1ms each, at 1MBaud

**CYCLE**

Cycle command has to be sent by broadcast, and will only be successful if the START command has been done at least once.

This command will start the cycle of replies from the SERVOs to the MASTER, one after the other by the order set on the START.

A reply to this command will be the Position of each servo, if successfully accomplished.

Master to Servo

Packet 0xAA 0x03 0x01 0x0FE 0x07 0xZZ

0xAA is the header

0x03 The Length

0x01 The Destination ID

0xFE Master ID

0x01 START request

0xZZ CRC

Servo1 to Master

Packet 0xAA 0x03 0xFE 0x01 0xnn 0xnn 0xZZ

0xAA is the header

0x03 The Length

0xFE The Destination ID

0x01 Slave ID

0xnn Position lsb

0xnn Position MSB

0xZZ CRC

Servo 2 to Master

Packet 0xAA 0x03 0xFE 0x02 0xnn 0xnn 0xZZ

…

Servo N to Master

Packet 0xAA 0x03 0xFE 0x0N 0xnn 0xnn 0xZZ

**OK**

Ok the command is the reply from the servo when a successful WRITE command is received and executed.

**NOK**

The NOK command is a reply from the SERVO when a bad CRC append, or a command is unsuccessful, due to erroneous data or out of bounds parameters.

**SYNC\_WRITE**

The syncronous write has to be sent by broadcast and allow to send data to several servos using one single packet to do it, SINCE the ADRESS and the LENGTH of data to be written to each servo is the SAME to all of them.

The packet composition is like following,

Master to Servo

Packet 0xAA 0xXX 0xFF 0xFE 0x83 0x0B 0x01 0xn1 0xn2 … 0xnn 0x020xn1 0xn2 … 0xnn … 0x0n 0xn1 0xn2 … 0xnn 0xZZ

0xAA is the header

0xXX The Length

0xFF The Destination BROADCAST ID

0xFE Master ID

0x83 SYNC WRITE request

0x0B Memory Address to start writing

0xnn Length of data to write in bytes

0x01 Servo ID 1

0xn1 data 1

0xn2 …

0xnn data n

0x02 Servo ID 2

0xn1 data 1

0xn2 …

0xnn data n

…

0x0n Servo ID n

0xn1 data 1

0xn2 …

0xnn data n

0xZZ CRC

Servo to Master

No reply.

**RESET**

The reset command can be sent either by regular ID or broadcast, and can be used to reset ALL the EEPROM data to the default values, including ID and BAUDRATE.

**Predefined values and respective ranges:**

**Memory organization**

The memory organization implemented is continuous between eeprom and ram as shown below, allowing this to read or write eeprom, ram or both with a single request command.

|  |  |  |
| --- | --- | --- |
| 0x00 | Reserved | Reserved |
| 0x01 | MY ID | READ / WRITE |
| 0x02 | CTL UP LIMIT | READ / WRITE |
| 0x03 | CTL DW LIMIT | READ / WRITE |
| 0x04 | KPp | READ / WRITE |
| 0x05 | Kip | READ / WRITE |
| 0x06 | KDp | READ / WRITE |
| 0x07 | KPv | READ / WRITE |
| 0x08 | KIv | READ / WRITE |
| 0x09 | KDv | READ / WRITE |
| 0x0A | UART BAUD | READ / WRITE |
| 0x0B | POSITION LSB | READ / WRITE |
| 0x0C | POSITION MSB | READ / WRITE |
| 0x0D | SPEED | READ / WRITE |
| 0x0E | TEMPERATURE LSB | READ |
| 0x0F | TEMPERATURE MSB | READ |

EEPROM

RAM

**MY ID**

This memory address contain the IDentification of the servomotor on the bus, used for example to identify the origin of it own messages to the bus or to calculate the delay on startup.

The ID can be changed by a broadcast, which we have to have in attention that all the servos on the bus will get the same ID, or can be changed by a packet addressed to the current ID with the new ID, to which the servo will reply with the new ID already. The Values to the ID must be within 1 and 253.

**Devices IDs**

The devices IDs have to be assigned with values within 1 and 254. Normally 254 is the value assigned to be the MASTER, but with this new protocol and packets composition, any ID can be MASTER, allowing also the support a Multi MASTER system on the BUS.

The address 255 is the common address of broadcasting.

POSITION

The position when reading returns the actual position of the servo, when writing sets the new Goal Position. The position must have values within

(0+CTL LW LIMIT) and (1023 – CTL UP LIMIT).

CTL UP/LW LIMIT

The Control upper limit is the margin in steps, the distance from the edge to consider as limit avoiding with this entering on the dead zone of the servo encoder.

Limited Rotation Full Rotation

Low Limit Upper Limit

0 Dead Zone 1023

**SPEED**

The speed when reading returns the actual speed of the servo when moving toward the goal position, when writing, sets the speed with which the servo will follow to the goal position. This speed is given in steps by 10ms, and we have 10 different speeds.

The values must be within 0 and 10

KPp, KIp and KDp

Constants to control the PID for the static positioning control loop.

Proportional, Integral and Derivate

OldError= Error

Error = GoalPosition-Position

ErrorAc = ErrorAc+OldError

DeltError = Error-OldError

Result = 1000 – (( KPp x Error) + (( KIp x ErrorAc)/100) + (KDp x DeltError) )

KPv, KIv and KDv

Constants to control the PID for the dynamic positioning control loop.

Proportional, Integral and Derivate

OldError= Error

Error = GoalPosition-Position

ErrorAc = ErrorAc+OldError

DeltError = Error-OldError

Result = 1000 – (( KPv x Error) + (( KIv x ErrorAc)/10) + (KDv x DeltError) )

**Proportional**

Give the *speed* toward the goal position

**Derivate**

Give the *slowing down* when reaching the goal point

**Integral**

Give the *error correction* around the goal position

Derivate

Integral

Proportional

**UART BAUD**

Baudrate to control the UART to the bus, once again not that after modifying the uart baudrate the reply to the command will be done with the new baudrate.

|  |  |
| --- | --- |
| UART BAUD | Value to set |
| 9600 | 207 |
| 57600 | 34 |
| 115200 | 16 |
| 250000 | 7 |
| 500000 | 3 |
| 1000000 | 1 |

**EEPROM Default values**

MY ID 0x01

CTL UP LIMIT 0x02

CTL LW LIMIT 0x02

KPp 0x28

KIp 0x01

KDp 0x0A

KPv 0x28

KIv 0x0A

KDv 0x00

UART BAUD 0x01

**TYPICAL CLOSED LOOP DATA FLUX**

|  |  |  |
| --- | --- | --- |
| MASTER \* Packets |  | SLAVE\*\* Packets |
| START (Broadcast) |  |  |
|  |  | PING (Servo 1) |
|  |  | PING (Servo 2) |
|  |  | … |
|  |  | PING (Servo N) |
|  | … |  |
| CYCLE (Broadcast) |  |  |
|  |  | POSITION (Servo 1) |
|  |  | POSITION (Servo 2) |
|  |  | … |
|  |  | POSITION (Servo N) |
|  | … |  |
| SYNC\_WR (Broadcast new positions and speeds) |  |  |

\* This can be the PC, Roboard, CM5, or more than one of them at the same time.

\*\* These are the Servomotors

By :

Ricardo Silva Marinheiro

14 February,2011