

Propeller Dynamixel AX-12 Command Reference

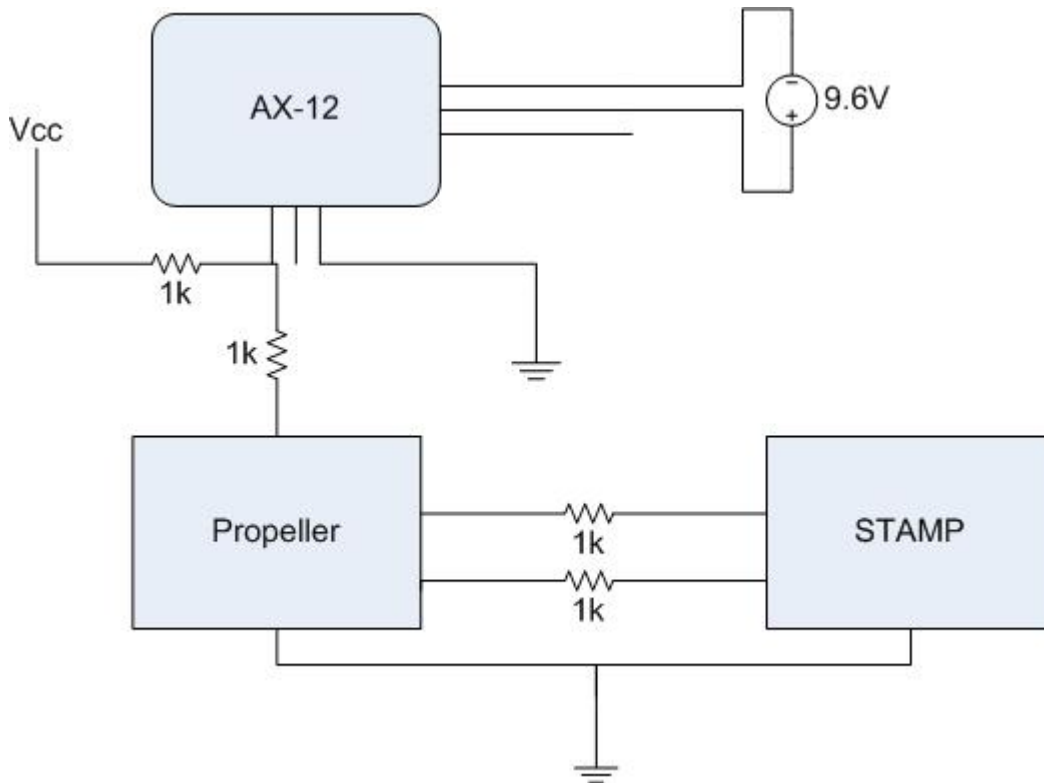
<i>Propeller Dynamixel AX-12 Command Reference</i> _____	1
Overview _____	2
Schematic _____	2
Simple PBasic Command Structure _____	3
Sending Data _____	3
Receiving data _____	3
Command Syntax _____	4
Ping: !AXPG _____	4
Read Data: !AXRD _____	4
Write Data: !AXWR _____	5
Reg Write: !AXRW _____	5
Action: !AXAC _____	6
Reset: !AXRS _____	6
Sync Write: !AZSW _____	6
Code Files _____	7
AX-12 Control Table Registers _____	8

Overview

The Dynamixel AX-12 bus operates at 1Mbps by default. 1Mbps is fairly quick for many microcontrollers so the software development team at CrustCrawler created an embedded solution using the Parallax Propeller. This solution allows microcontroller like the Parallax STAMP to send and receive data on an AX-12 bus without slowing down the bus.

Schematic

Add a 1k resistor between the STAMP and Propeller I/O pins. This is important because the STAMP is a TTL device and the Propeller is a CMOS device (5V and 3.3V). Place a 1 k resistor between the AX-12 signal line and Propeller I/O pin. Use another 1k resistor to pull the $\frac{1}{2}$ duplex serial bus high (3.3V) when the bus is in the idle state. Power the AX-12 with 7 to 10 volts. The AX-12 has an LED that flashes on when powered correctly.



Simple PBasic Command Structure

This document was created with the STAMP microcontrollers in mind so the examples are in PBasic. Please see your microcontroller documentation for command syntax.

These are the commands we'll be using:

!AXPG	Ping
!AXRD	Read Data
!AXWR	Write Data
!AXRW	Reg Write
!AXAC	Action
!AXRS	Reset
!AZSW	Sync Write

Sending Data

To send serial data with a STAMP use the SEROUT command. Both examples below ping the AX-12 with an ID of 5. The first SEROUT command is an example of a native AX-12 command. See the AX-12 manual. The second SEROUT command is an abbreviated command. We will be discussing the abbreviated commands in this document but it is good to know that you can send native AX-12 commands.

Note: PropTx is the output pin and Baud is the baud rate. The demo code uses pin 0 and 2400 baud. The "\$" means hexadecimal (Base 16) in PBasic I'll use the standard notation 0x05 for the remainder of this document.

```
SEROUT PropTx, Baud, [$FF, $FF, $05, $02, $01, $F7]
```

Or

```
SEROUT PropTx, Baud, ["!AXPG", $05]
```

You'll notice that the commands all start with "!AX". This is a construct that Parallax uses to signify the type of device being used; we decided to do the same. The next 2 bytes form the command instruction. Following the instruction ("!AXPG") are the instruction arguments. The Ping command uses one argument, the ID of the device you want to ping.

Receiving data

Pinging is not very useful without a response. We'll use the PBasic SERIN command to read the response packet from the AX-12 network.

```
SERIN PropRx, Baud, 20, Print, [STR serPre\4]
```

```
SERIN PropRx, Baud, 20, Print, [STR serStr\serPre(3)]
```

The SERIN construct is similar to the SEROUT command with a couple of extra arguments. The "20" is the amount of time to wait (timeout) for incoming data and "Print" is the sub routine to jump to if a timeout occurs. See the PBasic help file for

details. Why are there two SERIN commands and one SEROUT? Well, it has to do with the way the AX-12 response packets are setup.

This is an example response packet.

`0xFF` `0x$FF` `ID` `Length` `Error` `Parameter0...N` `Check Sum`

The segment `[STR serPre\4]` means get 4 bytes and put the bytes in the serPre array. Array element 3 contains the length byte which is the length of the rest of the packet. Therefore, `[STR serStr\serPre(3)]` gets the rest of the packet.

Command Syntax

Ping: !AXPG

Send

`!AXPG` `ID`

Response

`0xFF` `0x$FF` `ID` `Length` `Error` `Check Sum`

As discussed above the Ping command has one argument, the id of the AX-12 that you want to ping.

Sending the command

```
SEROUT PropTx, Baud, ["!AXPG", $05]
```

Reading the response

```
SERIN PropRx, Baud, 20, Print, [STR serPre\4]  
SERIN PropRx, Baud, 20, Print, [STR serStr\serPre(3)]
```

Read Data: !AXRD

The !AXRD instruction is a request to read X number of bytes from the AX-12 register table starting at register address Y.

Send

`!AXRD` `ID` `Start Address` `Number of Bytes`

Response

`0xFF` `0x$FF` `ID` `Length` `Error` `Parameter1` `Parameter2` `Parameter N` `Check Sum`

Sending the command

```
SEROUT PropTx, Baud, ["!AXRD", $05, $24, $08]
```

The Read Data command is requesting information from AX-12 with the ID 5. The example above is requesting that ID 5 send back 8 (0x08) bytes of contiguous control table data starting at address 36 (0x24). See Table 1

Reading the response

```
SERIN PropRx, Baud, 20, Print, [STR serPre\4]
```

```
SERIN PropRx, Baud, 20, Print, [STR serStr\serPre(3)]
```

Write Data: !AXWR

The Write Data command writes to the AX-12 control table.

Send

```
!AXRD ID Start Address Number of Bytes Byte1 Byte2 ByteN
```

Response

```
0xFF 0xFF ID Length Error Parameter1 Parameter2 Parameter N Check Sum
```

In the example below we're writing 1 byte starting at control table address 25. The value we're writing is 1 (the last number). This example turns on the LED on ID5.

Sending the command

```
SEROUT PropTx, Baud, ["!AXWD", $05, $19, $01, $01]
```

This code segment turns the LED off.

```
SEROUT PropTx, Baud, ["!AXWD", $05, $19, $01, $00]
```

Reading the response

```
SERIN PropRx, Baud, 20, Print, [STR serPre\4]
```

```
SERIN PropRx, Baud, 20, Print, [STR serStr\serPre(3)]
```

Reg Write: !AXRW

The Reg Write instruction is constructed like the Write Data instruction. The difference is Reg Write instructions are buffered until the Action command is received. Reg Write is useful for syncing multiple AX-12s that need to move simultaneously.

Send

```
!AXRW ID Start Address Number of Bytes Byte1 Byte2 ByteN
```

Response

```
None
```

Sending the command

SEROUT PropTx, Baud, ["!AXRW", \$05, \$19, \$01, \$01]

This code turns the LED on after the Action command is executed

Action: !AXAC

The action command is used to execute buffered data from the Reg Write command.

Send

!"!AXAC"

Response

None

Sending the command

SEROUT PropTx, Baud, ["!AXAC"]

Reset: !AXRS

Not implemented at this time.

Sync Write: !AZSW

The Sync Write instruction is used to control several AX-12 at the same time using one instruction. There is only one rule, you must write to the same control table registers for each Dynamixel.

Send

!AXSW ID # of AX-12s # of Bytes per AX-12 Starting address ID Byte1
Byte2 ByteN ID Byte1 Byte2 ByteN ...

Response

None

Sending the command

SEROUT PropTx, Baud, ["!AXSW", \$02, \$04, \$1E, \$05, \$00, \$02, \$00, \$02, \$06, \$00,
\$02, \$00, \$02]

The example above moves ID 5 to position 512 at the speed of 512 and also moves ID 6 to position 512 at the speed of 512. Be aware that when sending a word value (2 bytes) you always send the low byte first. In this case 512 decimal is 0x200 therefore the output is formatted \$05 \$00 \$02.

Code Files

Below are the files you need to get started. At the time of this writing the ½ duplex serial driver file was named DynamixelAsmBus_v0.34_03122008.spin. The name will most likely change in the future.

File Name	Description
DynaBus.spin	Main object file. This is the one that you load into the Propeller. It will load the other files.
DynamixelAsmBus_v0.34_03122008.spin	½ duplex assembly driver
FullDuplex.spin	Parallax serial driver
AX12Client.bs2	Demo code for the STAMP.

All source code files can be found on www.crustcrawler.com.

Happy Programming!

AX-12 Control Table Registers

Table 1

Address	Item	Access	Initial Value
0(0X00)	Model Number(L)	RD	12(0x0C)
1(0X01)	Model Number(H)	RD	0(0x00)
2(0X02)	Version of Firmware RD ?	RD	?
3(0X03)	ID	RD,WR	1(0x01)
4(0X04)	Baud Rate	RD,WR	1(0x01)
5(0X05)	Return Delay Time	RD,WR	250(0xFA)
6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
10(0x0A)	(Reserved) -		0(0x00)
11(0X0B)	Highest Limit Temperature	RD,WR	85(0x55)
12(0X0C)	Lowest Limit Voltage	RD,WR	60(0X3C)
13(0X0D)	Highest Limit Voltage	RD,WR	190(0xBE)
14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
16(0X10)	Status Return Level	RD,WR	2(0x02)
17(0X11)	Alarm LED	RD,WR	4(0x04)
18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
19(0X13)	(Reserved)	RD,WR	0(0x00)
20(0X14)	Down Calibration(L)	RD	?
21(0X15)	Down Calibration(H)	RD	?
22(0X16)	Up Calibration(L)	RD	?
23(0X17)	Up Calibration(H)	RD	?
24(0X18)	Torque Enable	RD,WR	0(0x00)
25(0X19)	LED	RD,WR	0(0x00)
26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
32(0X20)	Moving Speed(L)	RD,WR	0
33(0X21)	Moving Speed(H)	RD,WR	0
34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
36(0X24)	Present Position(L)	RD	?
37(0X25)	Present Position(H)	RD	?
38(0X26)	Present Speed(L)	RD	?
39(0X27)	Present Speed(H)	RD	?

40(0X28)	Present Load(L)	RD	?
41(0X29)	Present Load(H)	RD	?
42(0X2A)	Present Voltage	RD	?
43(0X2B)	Present Temperature	RD	?
44(0X2C)	Registered Instruction	RD,WR	0(0x00)
45(0X2D)	(Reserved) -	0(0x00)	
46[0x2E)	Moving	RD	0(0x00)
47[0x2F)	Lock	RD,WR	0(0x00)
48[0x30)	Punch(L)	RD,WR	32(0x20)
49[0x31)	Punch(H)	RD,WR	0(0x00)