AIMB Protocol v0.8.8

Microcontroller Communication Protocol Definition

18.06.2015

beCee Soft Art

André Spitzner

Content

[1. List of Commands 3](#_Toc422427106)

[1.1 TIMER - Timer Sensor 4](#_Toc422427107)

[1.2 MOVE – Engine Speed Command 5](#_Toc422427108)

[1.3 TEMP - Temperature Sensor 6](#_Toc422427109)

[1.4 STATE - Enable / Disable Sensor 7](#_Toc422427110)

[1.5 SPEED - Speed Sensor 8](#_Toc422427111)

[1.6 RVNFO - Robot Version Information 9](#_Toc422427112)

[1.7 LIGHT - Light Sensor 10](#_Toc422427113)

[1.8 ERROR - Robot Error Information 11](#_Toc422427114)

[1.9 WARNING - Robot Warning Information 12](#_Toc422427115)

[1.10 ENGINE - Engine Sensor 13](#_Toc422427116)

[1.11 DISTANCE - Distance Sensor 14](#_Toc422427117)

[1.12 BUMPER - Bumper Sensor 15](#_Toc422427118)

[1.13 ACS - ACS Sensor 16](#_Toc422427119)

[1.14 ACDC - Voltage Sensor 17](#_Toc422427120)

[1.15 HUMIDITY – Relative Humidity 18](#_Toc422427121)

[1.16 ANALOG – Analog Port Voltages 19](#_Toc422427122)

[1.17 ANALOGPLAIN – Analog Port Plain Value 20](#_Toc422427123)

[1.18 ELTIME – Elasped Time 21](#_Toc422427124)

[1.19 FREERAM – Free RAM Memory 22](#_Toc422427125)

[1.20 USEDRAM – Used RAM Memory 23](#_Toc422427126)

[1.21 PROGSIZE – Programm Size 24](#_Toc422427127)

[1.22 PRESSURE – Barometric Pressure 25](#_Toc422427128)

[1.23 COMPASS – 3 Axis Magnetometer Compass 26](#_Toc422427129)

[1.24 HALL – Hall Current Sensor 27](#_Toc422427130)

[1.25 DELAY – Sample Rate 28](#_Toc422427131)

[1.26 SENSORRATE – Sensor Measurement Rate 29](#_Toc422427132)

[1.27 VIBRATION – Vibration Sensor 30](#_Toc422427133)

[1.28 ACCEL – 3 Axis Accelerometer 32](#_Toc422427134)

[1.29 LRFC – Laser Range Finder Command 33](#_Toc422427135)

[1.30 UDSC – Ultrasonic Distance Sensor Command 35](#_Toc422427136)

[1.31 SERVOC – Servo Command 36](#_Toc422427137)

[1.32 DIGITAL – Digital Pin 37](#_Toc422427138)

[2. Needed Functions 38](#_Toc422427139)

[2.1 ConvertToVoltage 39](#_Toc422427140)

[2.2 MicroSecondsToCentimeters 40](#_Toc422427141)

[2.3 GetTemperatur 41](#_Toc422427142)

[2.4 GetVibration 42](#_Toc422427143)

[2.5 ReadUltrasonicSensor 43](#_Toc422427144)

[2.6 GetHumidity 44](#_Toc422427145)

[2.7 FreeRam 45](#_Toc422427146)

# List of Commands

**The commands are NMEA like styled for easy to use and understand.**

**Overview of commands:**

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Platform** |
| TIMER | Timer Sensor | RP6 |
| MOVE | Engine Speed Command | RP6 |
| TEMP | Temperature Sensor | Arduino, MS IoT |
| STATE | Enable / Disable Sensor | RP6 |
| SPEED | Speed Sensor | RP6 |
| RVNFO | Robot Version Information | Arduino, MS IoT |
| LIGHT | Light Sensor | RP6 |
| ERROR | Robot Error Information | RP6, Arduino, MS IoT |
| WARNING | Robot Warning Information | Arduino, MS IoT |
| ENGINE | Engine Sensor | RP6 |
| DISTANCE | Distance Sensor | RP6, Arduino |
| BUMPER | Bumper Sensor | RP6 |
| ACS | ACS Sensor | RP6 |
| ACDC | Voltage Sensor | RP6 |
| HUMIDITY | Relative Humidity | Arduino, MS IoT |
| ANALOG | Analog Sensors Voltages (A0 - A5) | Arduino, MS IoT |
| ANALOGPLAIN | Analog Sensors Values (A0 - A5) | Arduino, MS IoT |
| ELTIME | Elapsed Time For Executing | Arduino, MS IoT |
| FREERAM | Free Memory | Arduino |
| USEDRAM | Used Memory | Arduino |
| PRESSURE | Barometric Pressure | Arduino |
| COMPASS | 3 Axis Magnetometer Compass | Arduino |
| HALL | Hall Current Sensor | Arduino |
| DELAY | Regulates The Time Between Measurements(Sample Rate) | Arduino |
| SENSORRATE | Regulates The Measurement Rate Before Of A Sensor | Arduino |
| VIBRATION | Analog Vibration Sensor | Arduino |
| ACCEL | Accelerometer Sensor | Arduino |
| LRFC | Laser Range Finder Command | Arduino |
| UDSC | Ultrasonic Distance Sensor Command | Arduino |
| SERVOC | Servo Command | Arduino |
| DIGITAL | Digital Pin Value | MS IoT |

* 1. TIMER - Timer Sensor

1 2  
 | |  
Sentence: $--TIMER,1,300  
1: Timer Id  
2: Timer Value  
  
Direction: **In**

Function:

The command adjusts the time interval of data is sent by the robot.

For the RP6 robot the timer id 1-8 can be used.  
  
Timer Id Changes Default Value  
1 RVNFO 10000 ms  
2 ACDC 1000 ms  
3 LIGHT 1000 ms  
4 SPEED, ENGINE 1000 ms  
5 DISTANCE 1000 ms  
6 ACS 200 ms  
7 currently unused   
8 currently unused

* 1. MOVE – Engine Speed Command

1 2   
 | |   
Sentence: $--MOVE,1,100

1: Actuator Id  
2: Motor Speed  
  
Direction: **In**  
  
Notice (only RP6):   
The motor speed is default limited to 200 by the RP6 base code.  
  
Function:

Use actuator id = 1 for regulating speed of the engine on the left side and actuator id = 2 for the engine of the right side.  
All other values for actuator id will set engine power to 0 for both engines.

* 1. TEMP - Temperature Sensor

1 2 3  
 | | |  
Sentence: $--TEMP,1,27.25,°C

1: Sensor Id  
2: Temperature Value  
3: Unit  
  
Direction: **Out**

**Example:**

Protocol.WriteTemperature(A0);

* 1. STATE - Enable / Disable Sensor

1 2  
 | |  
Sentence: $--STATE,1,1

1: Sensor Id  
2: State 0=Disabled/1=Enabled  
  
Direction: **IN**

* 1. SPEED - Speed Sensor

1 2 3  
 | | |  
Sentence: $--SPEED,1,300,mm/s

1: Sensor Id  
2: Speed Value  
3: Unit  
  
Direction: **Out**

* 1. RVNFO - Robot Version Information

1 2 3 4  
 | | | |  
Sentence: $--RVNFO,Identifier,Version,Description,Vendor

1: Identifier  
2: Version  
3: Description  
4: Vendor  
  
Direction: **Out**

**Example:**

Protocol.WriteRVNFO("AIMB\_00", "AIMB v0.8.4", "AIMB Protocol Test", "beCee Soft Art");

* 1. LIGHT - Light Sensor

1 2  
 | |  
Sentence: $--LIGHT,1,300

1: Sensor Id  
2: Light Value  
  
Direction: **Out**

* 1. ERROR - Robot Error Information

1 2 3  
 | | |  
Sentence: $--ERROR,1,1,Description

1: Sensor Id  
2: Error Id  
3: Description  
  
Direction: **Out**

**Example:**

Protocol.WriteError(A0, 1, “Out Of Range”);

* 1. WARNING - Robot Warning Information

1 2 3  
 | | |  
Sentence: $--WARNING,1,1,Description

1: Sensor Id  
2: Warning Id  
3: Description  
  
Direction: **Out**

**Example:**

Protocol.WriteError(4, 1, “Engine Off By Distance”);

* 1. ENGINE - Engine Sensor

1 2   
 | |   
Sentence: $--ENGINE,1,200

1: Sensor Id  
2: Engine Value  
  
Direction: **Out**

* 1. DISTANCE - Distance Sensor

1 2 3  
 | | |  
Sentence: $--DISTANCE,1,300,mm/s

1: Sensor Id  
2: Distance Value  
3: Unit  
  
Direction: **Out**

**Example:**

Protocol.WriteUltrasonicDistance(7);

* 1. BUMPER - Bumper Sensor

1 2  
 | |  
Sentence: $--BUMPER,1,1

1: Sensor Id  
2: Bumper Value  
  
Direction: **Out**

* 1. ACS - ACS Sensor

1 2  
 | |  
Sentence: $--ACS,1,1

1: Sensor Id  
2: ACS Value  
  
Direction: **Out**

* 1. ACDC - Voltage Sensor

1 2 3  
 | | |  
Sentence: $--ACDC,1,8.05,V

1: Sensor Id  
2: Voltage Value  
3: Unit  
  
Direction: **Out**

* 1. HUMIDITY – Relative Humidity

1 2 3  
 | | |  
Sentence: $--HUMIDITY,1,8.05,%RH

1: Sensor Id  
2: Humidity Value, Relative in Percent  
3: Unit  
  
Direction: **Out**

Calculation for using HIH-4030/31 humidity sensor:

(Analog sensor voltage) - 0.958)/ 0.03068) = Relative humidity in %

**Example:**

Protocol.WriteHumidity(A2);

* 1. ANALOG – Analog Port Voltages

1 7 2 7 3 7 4 7 5 7 6 7  
 | | | | | | | | | | | |  
Sentence: $--ANALOG,1.10,V,0.00,V,2.10,V,1.70,V,1.20,V,1.00,V

1: Voltage on Analog A0  
2: Voltage on Analog A1  
3: Voltage on Analog A2  
4: Voltage on Analog A3  
5: Voltage on Analog A4  
6: Voltage on Analog A5  
7: Unit  
  
Direction: **Out**

**Notice:**

When no sensor is connected to an analog port, the value of this port is not zero “0”. The value ranges between 2.1 and 2.5 volt.

**Example:**

Protocol.WriteAnalog();

* 1. ANALOGPLAIN – Analog Port Plain Value

1 2 3 4 5 6   
 | | | | | |   
Sentence: $--ANALOGPLAIN,110,40,210,170,120,100

1: Value on Analog A0  
2: Value on Analog A1  
3: Value on Analog A2  
4: Value on Analog A3  
5: Value on Analog A4  
6: Value on Analog A5

Direction: **Out**

**Notice:**

When no sensor is connected to an analog port, the value of this port is not zero “0”. The value ranges between 450 and 500.

**Example:**

Protocol.WriteAnalogPlain();

* 1. ELTIME – Elasped Time

1 2 3  
 | | |  
Sentence: $--ELTIME,1,140,ms

1: Sensor Id  
2: Elapsed Time  
3: Unit (micro / ms / s)  
  
Direction: **Out**

**Example:**

Protocol.StartElapsedTime();

// Do Something  
Protocol.WriteElapsedTime();

* 1. FREERAM – Free RAM Memory

1 2  
 | |  
Sentence: $--FREERAM,1124,Byte

1: Free RAM   
2: Unit (Byte)   
  
Direction: **Out**

**Example:**

Protocol.WriteFreeRam();

* 1. USEDRAM – Used RAM Memory

1 2  
 | |  
Sentence: $--USED,1124,Byte

1: Used RAM   
2: Unit (Byte)   
  
Direction: **Out**

**Example:**

Protocol.WriteUsedRam();

* 1. PROGSIZE – Programm Size

1 2  
 | |  
Sentence: $--PROGSIZE,14124,Byte

1: Program size   
2: Unit (Byte)   
  
Direction: **Out**

**Example:**

Protocol.WriteProgramSize();

* 1. PRESSURE – Barometric Pressure

1 2 3  
 | | |  
Sentence: $--PRESSURE,19,0.914,Bar

1: Sensor Id   
2: Value  
3: Unit (Bar)   
  
Direction: **Out**

Notice:

Works with BMP085 barometric pressure sensor on I2C bus.

**Example:**

Protocol.WritePressure();

* 1. COMPASS – 3 Axis Magnetometer Compass

Sentence:

1 2 3 4 5 6 7 8 9  
 | | | | | | | | |  
$--COMPASS,19,142,-83,-518,130.64,-76.36,-476.56,5.75,329.69

1: Sensor Id   
2: Raw X Axis Value  
3: Raw Y Axis Value  
4: Raw Z Axis Value  
5: Scaled X Axis Value  
6: Scaled Y Axis Value  
7: Scaled Y Axis Value  
8: Heading Value  
9: Radians Degrees Value   
  
Direction: **Out**

Notice:

Works with HMC5883L triple axis magnetometer on I2C bus.

**Example:**

Protocol. WriteCompassHMC5883L();

* 1. HALL – Hall Current Sensor

1 2 3  
 | | |  
Sentence: $--HALL,19,0.176,A

1: Sensor Id   
2: Value  
3: Unit (A = Ampere)   
  
Direction: **Out**

Notice:

Works with ACS712 Hall Effect based linear current sensor.

**Example:**

Protocol.WriteHall(A0);

* 1. DELAY – Sample Rate

1  
 |  
Sentence: $--DELAY,1000

1: Delay (Milliseconds)

Direction: **In**

This command sets the delay time between measurements.

Notice:

Real delay time is calculated by “delay time – cpu cycle time”. So should the sample rate value larger than the needed execution time of the program for one cycle.

The command can use with command SENSORRATE.

* 1. SENSORRATE – Sensor Measurement Rate

1  
 |  
Sentence: $--SENSORRATE,150

1: Sensor Measurement Rate (Count of measurements)

Direction: **In**

This command sets the measurement rate for analog and digital sensors. The command value is calculated by average of measurements.

Notice:

A longer measurement rate has the effect that the duration of the execution time of the program will be greater. But a larger measurement rate value smoothens the value.

* 1. VIBRATION – Vibration Sensor

1 2 3  
 | | |  
Sentence: $--VIBRATION,19,0.176,A

1: Sensor Id   
2: Value  
3: Unit (V = Volt)   
  
Direction: **Out**

Notice:

Works with piezo based vibration sensor.

**Example:**

Protocol.WriteVibration(A0);

* 1. ACCEL – 3 Axis Accelerometer

Sentence:

1 2 3 4 5 6 7   
 | | | | | | |   
$--ACCEL,19,142,-83,-518,130.64,-76.36,-476.56

1: Sensor Id   
2: Raw X Axis Value  
3: Raw Y Axis Value  
4: Raw Z Axis Value  
5: Scaled X Axis Value  
6: Scaled Y Axis Value  
7: Scaled Y Axis Value  
Direction: **Out**

Notice:

Works with ADXL345 triple axis accelerometer on I2C bus.

**Example:**

Protocol.WriteAccelerometerADXL345();

* 1. LRFC – Laser Range Finder Command

Sentence:

1 2

| |   
$--LRFC,19,R\*checksum

1: Sensor Id   
2: Command Character  
Direction: **In**

Notice:

Works with Parallax Laser Range Finder (Product Id: 28044) on Serial Port.

Dommand character can be:

* “R” = Measure distance
* “U” = Initialize sensor
* “A” = Scan full range (2 Axes)

**Example:**

LaserrangeFinder.Initialize(&Serial3,distanceOffset);

Double distance = LaserrangeFinder. MeasureDistanceCMF();

* 1. UDSC – Ultrasonic Distance Sensor Command

Sentence:

1 2

| |   
$--UDSC,19,R\*checksum

1: Sensor Id   
2: Command Character  
Direction: **In**

Notice:

Dommand character can be:

* “R” = Measure distance

**Example:**

* 1. SERVOC – Servo Command

Sentence:

1 2

| |   
$--SERVOC,1,100\*checksum

1: Sensor Id   
2: Degrees 0 – 180 (Integer)  
Direction: **In**

Notice:

Works with standard servo class implementation.

**Example:**

* 1. DIGITAL – Digital Pin

Sentence:

1 2

| |   
$--DIGITAL,1,100\*checksum

1: Sensor Id (Pin)  
2: Value  
Direction: **Out**

Notice:

Get value from digital pin.

**Example:**

# Needed Functions

Listening of helper functions.

* 1. ConvertToVoltage

Converts value from analog sensor to voltage value.

float AIMBProtocol::ConvertToVoltage(float fValue)

* 1. MicroSecondsToCentimeters

Converts time value from ultrasonic sensor to unit of length.

long AIMBProtocol::MicroSecondsToCentimeters(long lMicroseconds)

* 1. GetTemperatur

Calculation for using LM35 (LM35CZ) temperature sensor on analog port:

(Analog sensor value) / 2 = Temperature in °C

float AIMBProtocol::GetTemperatur(int iAnalogPort)

* 1. GetVibration

Calculation for using piezo vibration sensor on analog port:

float AIMBProtocol::GetVibration(int iAnalogPort)

* 1. ReadUltrasonicSensor

Calculation for using “Seed Ultrasonic” sensor connected to digital port.

long ReadUltrasonicSensor(int iDigitalId)

* 1. GetHumidity

Calculation for using HIH-4030/31 temperature sensor:

(Analog sensor voltage) - 0.958)/ 0.03068) = Relative humidity in %

float AIMBProtocol::GetHumidity(int iAnalogPort)

* 1. FreeRam

Calculate free RAM:

int AIMBProtocol::FreeRam();