AVA BUS OVERVIEW

What is Modbus?

Modbus is a serial communication protocol developed by Modicon published by Modicon® in 1979 for use with its programmable logic controllers (PLCs). In simple terms, it is a method used for transmitting information over serial lines between electronic devices. The device requesting the information is called the Modbus Master and the devices supplying information are Modbus Slaves. In a standard Modbus network, there is one Master and up to 247 Slaves, each with a unique Slave Address from 1 to 247. The Master can also write information to the Slaves. Find more info here: http://www.simplymodbus.ca/FAQ.htm

What is CAN Bus?

In short, the Controller Area Network (CAN) is a standard used to allow Electronic Control Units (ECUs) to communicate in an efficient manner without a central computer. Messages are broadcast in a system that requires very little physical wiring making CAN Bus low cost, robust and efficient.

Can you offer Profi-Bus?

Our engineers are currently working on Profi-Bus capabilities for the AVA actuators and we hope to have further update on this later in year. Keep checking www.avacuators.co.uk for more information about our progress.

Which Actuators can work with BUS systems and how?

Our AVA Smart actuators are capable of being supplied compatible with the systems as detailed above, Modbus and Canbus are currently available with our AVA 20, 110, 200 and 400Nm actuators. Other models are being added to the programme. The actuators are available as BUS Smart On Off or BUS Smart Modulating. The following pages will give you an overview of how the Smart actuators Firmware can be used to calibrate and set-up our actuators to your BUS system. The Smart intuitive menu system allows the user to customise and setup actuators specific to your requirement. When we say our actuators are Smart, we really mean it and our BUS actuator series further

Example of BUS system
**RTU mode:** Host demand data message formats

ADDR FUNC LEN D0...Dn-1 CRC
[ADDR]: Slaver address
[FUNC]: Function code: [XX]
[LEN]: Length of [D0...Dn-1] [YY]
[D0...Dn-1]: Information content LEN x 8bit [DAT0...DATn]
[CRC]: CRC16 code, host calculation leads to CRC code [CRCL CRCH]

Notice: The function formats data listed in this document are all hexadecimal.

### Distribution details of register function

<table>
<thead>
<tr>
<th>Register Addr (hexadecimal)</th>
<th>Name</th>
<th>Read-write state</th>
<th>Data length</th>
<th>Data content</th>
<th>Detailed account</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 01</td>
<td>Actuator Status</td>
<td>R</td>
<td>1</td>
<td>Specific data as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xAA</td>
<td>actuator acting command_ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xBB</td>
<td>actuator acting command_OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xA0</td>
<td>actuator acted ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xB0</td>
<td>actuator acted OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xBA</td>
<td>actuator acting command_BRAKE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xEA</td>
<td>actuator error status</td>
</tr>
<tr>
<td>00 02</td>
<td>Reserved</td>
<td>R</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00 03</td>
<td>Valve work mode</td>
<td>R</td>
<td>1</td>
<td>Specific data as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x22</td>
<td>two way mode</td>
</tr>
<tr>
<td>00 04</td>
<td>Controlling command</td>
<td>R/W</td>
<td>1</td>
<td>Specific data as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x01</td>
<td>command-on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x07</td>
<td>command-off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xBA</td>
<td>command-brake</td>
</tr>
<tr>
<td>00 05</td>
<td>Delay feedback time</td>
<td>R/W</td>
<td>1</td>
<td>0x01~0x32</td>
<td>1~50ms (default: 2ms)</td>
</tr>
<tr>
<td>00 06</td>
<td>Actuator Bus_ID</td>
<td>R</td>
<td>1</td>
<td>0x01~0xFC</td>
<td>bus ID (1~252)</td>
</tr>
<tr>
<td>00 00-00 0E</td>
<td>Production Date</td>
<td>R</td>
<td>1</td>
<td>YY YY MM DD, 0X140D 0X041C =&gt;2013-04-28</td>
<td></td>
</tr>
<tr>
<td>00 11-00 13</td>
<td>Use times</td>
<td>R</td>
<td>4</td>
<td>0xOOLL, 0xHHMM, Use times=LL+256<em>MM+65535</em>HH</td>
<td></td>
</tr>
<tr>
<td>00 14</td>
<td>Accurate position</td>
<td>R</td>
<td>2</td>
<td>0xZHDL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ZL=A%256, ZH=A/256, The opening of actuator=(A/10)%</td>
<td></td>
</tr>
<tr>
<td>00 21</td>
<td>Fault status</td>
<td>R</td>
<td>1</td>
<td>0x00</td>
<td>No fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xEA</td>
<td>Fault Status Clear the changed command</td>
</tr>
</tbody>
</table>
Join the orange and grey revolution

AVA BUS OVERVIEW

【Communicational function code between host and RS485 bus valve】

03/04  Read data out of valve register, even several continuous data is permitted.
06    Write single byte data in valve register

【Command formats of reading valve operation sent by host】

<table>
<thead>
<tr>
<th>Valve.Addr</th>
<th>03/04</th>
<th>00</th>
<th>XX</th>
<th>00</th>
<th>YY</th>
<th>CRCL</th>
<th>CRCH</th>
</tr>
</thead>
</table>

Within: Valve.Addr means valve bus address of reading operation;

03/04 means reading valve register;

00 XX means initial address to be read out (XX is only a 19-byte address from 0x01 to 0x13);

00 YY means (YY-XX+1) data read from 00 XX to (00 XX+00 YY) in register;

CRCL CRCH means two-byte CRC check code sent by host

【Data formats of reading valve to operate valve returning accurately】

<table>
<thead>
<tr>
<th>Valve.Addr</th>
<th>03/04</th>
<th>YY'2</th>
<th>DAT1</th>
<th>DAT2</th>
<th>...</th>
<th>DATYY</th>
<th>CRCL</th>
<th>CRCH</th>
</tr>
</thead>
</table>

Within: Valve.Addr means returning data of valve bus address after receive reading operation;

03/04 means reading operation of valve register;

YY means having read-out YY'2 byte data;

DAT1, DAT2...DATYY are read-out data;

CRCL CRCH means two-byte CRC check code calculated by valve MCU.

【Command formats of single-byte writing valve operation sent by host】

<table>
<thead>
<tr>
<th>Valve.Addr</th>
<th>06</th>
<th>00</th>
<th>XX</th>
<th>00</th>
<th>DAT</th>
<th>CRCL</th>
<th>CRCH</th>
</tr>
</thead>
</table>

Within: Valve.Addr means valve bus address to be written;

06 means writing valve register;

00 XX means register address in need of writing in register (only writable register is permitted);

00 DAT is write-in data in demand;

CRCL CRCH means two-byte CRC check code calculated by host.

【Data formats of single-byte writing valve to operate valve returning accurately】

<table>
<thead>
<tr>
<th>Valve.Addr</th>
<th>06</th>
<th>00</th>
<th>XX</th>
<th>00</th>
<th>DAT</th>
<th>CRCL</th>
<th>CRC</th>
</tr>
</thead>
</table>

Within: Valve.Addr means returning data of valve bus address after receive writing operation;

06 means writing valve register operation;

CRCL CRCH means two-byte CRC check code calculated by valve MCU.

【Data formats of valve returning on inaccurate operation】

<table>
<thead>
<tr>
<th>Valve.Addr</th>
<th>Function code+0x80</th>
<th>Error Code</th>
<th>CRCL</th>
<th>CRCH</th>
</tr>
</thead>
</table>

Within: Valve.Addr means valve bus address on error operation;

Function code+0x80 means inaccurate return;

Position means having returned valve position;

Error code: means inaccurate return, details as follow:

#define Err_FunCode 0x65 //101
#define Err_Register.Addr 0x66 //102
#define Err_Data.OutRange 0x67 //103
#define Err_Device 0x68 //104
#define Err_NOOMD 0x09 //201
#define Err_Modbus_Param 0x08 //203
Read out valve data which are written in register

It's necessary to realize bus address of valve to be operated before writing register operation in valve (Valve.Addr).

※ Read out valve bus address code without preconditions(Valve.Addr); (Notice: This command is only suitable for single valve suspended in bus.)

Send command: FD 5F 00 01 00 01 11 FB
Correct return: Valve.Addr 5F 00 01 00 01 CRCL CRCH
Error return: Valve.Addr DF 01 Error code CRCL CRCH

1. Command formats of reading out single byte data

Send format: Valve.Addr 03/04 00 XX 00 01 CRCL CRCH
   Within: Valve.Addr means valve bus address on present operation;
   03/04 means reading valve register operation;
   00 XX means register address to be read out (XX could only be 0x01-0x13);
   00 01 means having read out a byte data;
   CRCL CRCH means two-byte CRC check code.

When Valve.Addr = 0, null data return;
When Valve.Addr ≠ 0, valve corresponded in address code could return;

When the returning code is correct, it will get: Valve.Addr 04 02 DATL DATL CRCL CRCH
   Within: Valve.Addr means returning data of valve bus address after receive reading operation;
   DATL, DATL are read-out data;
   CRCL CRCH means two-byte CRC check code calculated by valve MCU.

【Example】Query valve addr. 01 state:
   Host sends: 01 04 00 02 00 01 90 0A
   Valve returns: 01 04 02 00 64 B8 DB (Bus addr.is 01 valve is in open position)

2. Command formats of reading out continuous multibyte data;

Send format: Valve.Addr 04 00 XX 00 YY CRCL CRCH
   Within: Valve.Addr means valve bus addr. at present operation;
   00 XX means reading out initial addr. from several continuous registers, (XX can only be 0x01-0x18);
   00 YY means the numbers of read-out registers;
   CRCL CRCH means two-byte CRC check code.

When Valve.Addr = 0, null data return;
When Valve.Addr ≠ 0, valve corresponded in address code could return;

When the returning code is correct, it will get: Valve.Addr 04 YY*2 DAT1 DAT2…DATn=2 CRCL CRCH
   Within: Valve.Addr means the returning data of valve bus address after receiving read operation;
   YY*2 means the numbers of bytes read out from valve code;
   DAT1, DAT2…DATn=2 are read-out data;
   CRCL CRCH means two-byte CRC check code calculated by valve MCU.

【Example 1】Query bus addr. 00 12 to 00 15;
   Host sends: 01 04 00 12 00 04 51 CC
   Valve returns next: 01 04 08 00 14 00 00 04 00 10 20 05
AVA BUS OVERVIEW

【Example 2】Query used time of bus addr.01 valve:
Host sends: 01 04 00 16 00 03 51 CF
Repeater returns first: 01 04 01 E5 80 02
Valve returns next: 01 04 03 05 00 00 E0 CF (means bus addr. 01 valve has used 5 times)

Ⅲ Modify the data written in valve register

Only the writable register data can be modified
1. Command formats of modifying Valve_Addr1 to Valve_Addr2 :
Send formats: Valve_Addr1 06 00 06 00 Valve_Addr2 CRCL CRCH
   Within: Valve_Addr1 means valve bus addr.at present operation;
   Valve_Addr2 means modified valve bus addr. ;
   CRCL CRCH means two-byte CRC check code.
   When Valve_Addr =0 , null data return;
   When Valve_Addr=0 , valve corresponded in address code could return;
When the returning code is correct, it will get: Valve_Addr1 06 01 Valve_Addr2 CRCL CRCH

【Example】Modify bus addr. 01 to bus addr. 02:
Host send : 01 06 00 06 00 02 E8 0A
Valve returns: 01 06 00 06 00 02 E8 0A (means having succeed to modify bus addr. 01 to bus addr. 02)

2. Control valve action
Send format: Valve_Addr 06 00 04 00 XX CRCL CRCH
   Within: Valve_Addr means valve bus addr.at present operation;
   00 04 means register which memoried valve controlling command.
   00 XX means controlling command code. Please refer to the following chart about the details;
   CRCL CRCH means two-byte CRC check code.
   When Valve_Addr =0 , null data return;
   When Valve_Addr=0 , valve corresponded in address code could return;
When the returning code is correct, it will get: Valve_Addr 06 00 04 00 XX CRCL CRCH

【Example】Bus addr.01 valve controlling command as follows:

<table>
<thead>
<tr>
<th>Control/Command</th>
<th>Send Command</th>
<th>Valve return</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>close valve</td>
<td>01 06 00 04 00 00 C8 0B</td>
<td>01 06 00 00 00 C8 0B</td>
<td></td>
</tr>
<tr>
<td>open valve</td>
<td>01 06 00 04 00 64 C9 E0</td>
<td>01 06 00 00 00 64 C9 E0</td>
<td></td>
</tr>
<tr>
<td>open30°</td>
<td>01 06 00 04 00 1E 48 03</td>
<td>01 06 00 04 00 1E 48 03</td>
<td>peculiar to 4-position 2-way</td>
</tr>
<tr>
<td>open60°</td>
<td>01 06 00 04 00 3C C8 1A</td>
<td>01 06 00 04 00 3C C8 1A</td>
<td>peculiar to 4-position 2-way</td>
</tr>
<tr>
<td>open to B33</td>
<td>01 06 00 04 00 B3 89 BE</td>
<td>01 06 00 04 00 B3 89 BE</td>
<td>peculiar to 3-position 3-way</td>
</tr>
<tr>
<td>brake</td>
<td>01 06 00 04 00 BA 49 B8</td>
<td>01 06 00 04 00 BA 49 B8</td>
<td></td>
</tr>
</tbody>
</table>