DAM-Series

User Manual User's Manual

Overview:

DAM module is a new generation data acquisition and control system based on modular embedded systems. Adopting standard DIN35 rail mounting package is simple, flexible respond to various field applications. They establish networks through RS485, RS422/RS232, Ethernet and other means of communication.

DAM-804x is the analog input series module, can simultaneously capture four differential or single-ended signal. The acquisition accuracy is up to 24 bit. Widely used to collecting various voltage and current signals on industrial site. Such as signals from a variety of sensors, transmitters, thermocouple can be gathered through it.

DAM-804x is with galvanic isolation technology and watchdog technology, effectively guaranteeing safe and reliable equipment operation.

Product Features:

- embedded real-time operating system
- Input signal range: ± 20mA, ± 10V, ± 5V ± 2.5V, ± 1V, ± 200mV, ± 100mV, ± 50mV
- Input Channels: 4 differential or four single-ended
- AD converter resolution: 24 Bit
- Measuring Accuracy: ± 0.05% (typical)
- Sampling rate: 20 times / sec (all channels)
- Wide range power supply: 8 - 40V DC
- Address / Baud rate / range configurable by the user
- Supports MODBUS RTU protocol
- ± 15KV ESD protection
- Isolation voltage: 3000V DC
- Operating temperature range: -40 ℃ - 85 ℃
- Industrial grade plastic shell, the standard DIN35 rail mounting

Products Application:

- Remote monitoring and data acquisition
- Intelligent Building Control
- Security Products and Security Project
- Industrial control on site
- Warehousing and monitoring
- Medical, industrial product development
- Food and beverage industry
- Packaging and Material Transfer
- Electronics Manufacturing
- Smart home system

Model NO.:	Function type
DAM-802x	General 2 channel analog input
DAM-804x	General 4 channel analog input
DAM-808x	General 8 channel analog input
DAM-902x	Multi-range 2 channel analog input
DAM-904x	Multi-range 4 channel analog input
DAM-908x	Multi-range 8 channel analog input

1

1. ASCII Custom ASCII Protocol

DAM module support custom ASCII protocol, user can easily read the measurement data and configuration module parameters through simple ASCII commands, such as address (0x00 - 0xFF), baud rate (300bps, 600bps, 1200bps, 2400bps, 4800bps, 9600bps , 19200bps, 38400bps, 57600bps, 115200bps), checksum status, open or close the channel and so on. When conditions permit, user can also calibrate the module on field through ASCII commands. Its instruction set compatible with the ADAM modules.

When the controller is in communication on the Modbus bus in ASCII protocol mode , a message is divided into two each 8Bit ASCII characters transmitted (eg send digital 58, 5 and 8 will transmit the corresponding ASCII codes 0x35 and 0x38), which the main advantage of such mode is to allow the time interval between characters up to 1 second, and an error does not occur.

ASCII protocol format of each byte:

- ◆ Coding system: hexadecimal, ASCII characters 0-9, A-F
- ◆ Data bits: 1 start bit, 8 data bits, 1 stop bit, no parity
- ◆ Error Checking: Longitudinal Redundancy Check

1.1 ASCII Command Syntax

ASCII command consist of a series of characters, including Delimiter, module address, keyword/variable, checksum (optional) and terminator (cr).DAM module doesn't support broadcast address, so host computer only can communicate with one DAM module at one time.

Command format：**(delimiter character)(address)(command)[data][checksum](cr)**

Command Form	Function/implication	length
delimiter character	Delimiter character, stands for start of the set command, have #/$/%	1 byte

address	Module address, Range is from 0x00 to 0xFF	2 byte
command	command keywords or variable	Decided by the command
data	commands necessary operand	Decided by the command
checksum	Checksum (optional)	2 byte
cr	Terminating character（Enter Key，value is0x0D）	1 byte

Form 1.1 ASCII Command Syntax Definition

When enable the checksum, it takes up 2 byte. Command and response both requires additional checksum feature. Checksum is used to check all the input command to help user find the communication errors from host computer to the module, the module to the host computer. Checksum is placed after the command or response character, before Carriage return character.

Computing method: Demand the sum of the ASCII number value of all the character send, then and the hexadecimal number 0xFF. The result is two hexadecimal digits.

Command Set Example:

Disable checksum

User Command **$012(cr)**

Module Response **!01000700 (cr)**

Enable checksum

User Command **$012B7 (cr)**

Module Response **!01000740AD (cr)**

Explain: $ 012 is the command to read the module configuration parameter, $ is the delimiter, 01 is the module address, 2 is the command keyword.

Hexadecimal value：‘$’ = 0x24 ‘0’ = 0x30 ‘1’ = 0x31

Checksum：B7=(0x24+0x30+0x31+0x32) AND 0xFF

Hexadecimal value：‘**!**’ = 0x21 ‘0’ = 0x30 ‘1’ = 0x31 ‘4’=0x34 ‘7’ = 0x37

Checksum：AD=(0x21+0x30+0x31+0x30+0x30+0x30+0x37+0x34+0x30) AND 0xFF

ASCII Command Set:

Command Syntax	Function	Section index	Remark
#AA	Read all channel data	1.2.1	
#AAN	Read channel N data	1.2.1	
%AANNTTCCFF	Parameter configuration	1.2.2	Effective only under configuration mode
$AA2	Read configuration state	1.2.3	
$AA0N	Channel N gain calibration	1.2.4	Effective only under configuration mode
$AA1N	Channel N offset calibration	1.2.5	Effective only under configuration mode
$AA6	Read channel status	1.2.6	
$AA8	Read measuring range	1.2.7	
$AA5xx	Enables or Disables channel	1.2.8	Effective only under configuration mode
$AAM	Read module name	1.2.9	
$AAV	Read firmware version	1.2.10	
$AAPx	Set communication protocol	1.2.11	Effective only under configuration mode
$AA7xx	Set measuring range	1.2.12	Only the multi-range module support the command， effective only under configuration mode

Form1.2 ASCII Command Set

1.2.1 Read Measuring data command

#AA

Name：Read all channel data command

Syntax：#AA(cr)

Delimiter

AA module address，range is from 00H to FFH

(cr) is the terminating character, carriage return (0Dh).

Response：>(data)(cr)；When the corresponding address module does not exist, non-response value

Example:

#01(cr) (Four channel analog input module)

Response：>+1.2345+1.2300+1.5780+1.700(cr)

Expain：> is delimiter，+ stands for the polarity of the measured data，(cr) is the terminating character.

#AAN

Name: Read channel N command

Syntax: # AAN (cr)

Delimiter

AA module address range is 00H to FFH

N indicates the channel number to be read (two-channel product is 0 or 1, the four-channel product is 0-3, eight-channel product is 0-7, sixteen-channel product is 0 to F)

(cr) carriage return character (0DH)

Normal Response:> (data) (cr)

Abnormal Response:? AA (cr) Channel number error or incorrect command length

Explain: > normal response delimiters; ? Abnormal response delimiter; data: is the return data; (cr) is the carriage return character; when the corresponding address of the module does not exist, no response.

Example: # 011 (cr)

Module response :> +2.5000 (cr)

Explain: # 011 is the channel data of read channel 1 with address 01H; the return value is +2.5000

1.2.2 Parameter Configuration Command

% AANNTTCCFF

Name: Parameter configuration command

Syntax: % AANNTTCCFF (cr)

% Delimiter

AA module address range is from 00H to FFH

NN module new address, range is from 00H to FFH

TT unused, reserved. Set to 00

CC baud rate code, the range is 0-9

FF checksum status, data format

(Cr) is the terminating character, carriage return (0Dh).

Normal Response:! NN (cr)

Exception Response:? AA (cr)

Explain:! Normal response delimiter; ? Abnormal response delimiter; (cr) carriage return character

CC value represents for serial baud rate, its corresponding baud rate as follow form:

Baud rate code（CC）	Baud Rate
01	300bps
02	600bps
03	1200bps
04	2400bps
05	4800bps
06	9600bps
07	19200bps
08	38400bps
09	57600bps
00	115200bps

Form 1.3 Baud rate codes comparison code

FF indicates checksum status and data format, its means as follow:

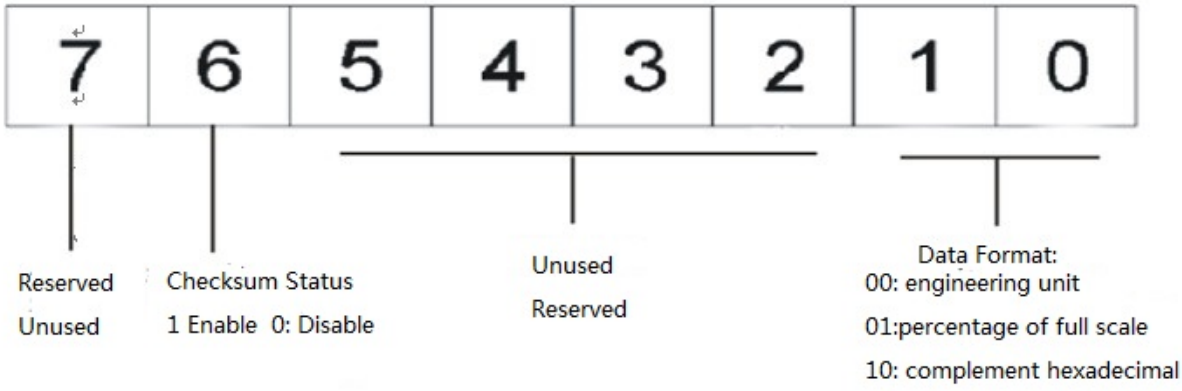

Figure 1.1 Setting bits bit definition

Explain:

Bit7 Reserved, set to 0

Bit6 Checksum status bits, 1 means enable checksum; 0 disables checksum

Bit5-Bit2 Reserved, set to 0

Bit1-Bit0 data format 00: Engineering Unit

01: Percentage of full scale (% of FSR)

10: Two's complement of hexadecimal

Example 1:

% 0123000700 (cr)

Module response: ! 23 (cr)

Explain: This command is to configure the module address 01H to 23H, baud rate 19200bps, data format to engineering unit, disable checksum.

Example 2:

% 0108000602 (cr)

Module response: ! 08 (cr)

Explain: This command is to configure the module address 01H to address 08H, baud rate 9600bps; data format to hexadecimal complement, checksum is disabled.

Example 3:

% 0109000841 (cr)

Module response: ! 08 (cr)

Explain: This command is to configure the module address 01H to 09H, baud rate 38400bps, the data format is percentage of full scale, enabled checksum.

1.2.3 Read Configuration Status Command

$ AA2

Name: Read Configuration Status command

Syntax: $AA2 (cr)

$ Delimiter

AA module address, range is from 00H to FFH

2 command keyword, here must be 2, otherwise you may perform other operations

(cr) is the terminating character, carriage return (0Dh)

Normal Response: ! AATTCCFF (cr)

Abnormal Response: ? AA (cr) command is entered incorrectly

Explain:! Normal response delimiter; ? Abnormal response delimiter; (cr) carriage return terminator. Normal response specification see chapter 1.2.2.

Example:

$ 082 (cr)

Module response: ! 08000702 (cr)

Explain: Command $ 082 (cr) is read address 08H module configuration status; ! Is the delimiter; 08 is the module address; 07 indicates the baud rate is 19200bps; 02 indicates that the data in hexadecimal format complement; (cr) is a carriage return character

1.2.4 Gain Calibration Command

$ AA0N

Name: gain calibration command

Syntax: $ AA0N (cr)

$ Delimiter

AA module address, range is from 00H to FFH

0 command keyword, here must be 0, otherwise you may perform other operations

N indicates the channel number to calibrate

Normal Response: ! AA (cr) successful implement a calibration

Abnormal Response: ? AA (cr) command is entered incorrectly

Explain:! Normal response delimiter;? Abnormal response delimiter; (cr) carriage return character

Example 1:

$ 0103 (cr)

Module response:! 01 (cr)

Explain: The $ 0103 (cr) is to gain calibrate channel 3 of module address with 01H; $ is the delimiter; 01H is the module address; 0 is a command keyword; 3 is the channel number; (cr) is a carriage return character

Example 2:

$ 0602 (cr)

Module response:! 06 (cr)

Explain: The $ 0602 (cr) is gain calibration of channel 2 with the module address 06H; $ is the delimiter; 06H is the module address; 0 is the command keyword; 2 is the channel number; (cr) is a carriage return character

1.2.5 Offset Calibration Command

$AA1N

Name: Offset Calibration Command

Syntax: $AA1N(cr)

$ Delimiter

AA Module Address, rang is from 00H to FFH

1 command keywordw, here must be 1, or other operation be performed

N represents the channel number need to be calibrated

Normal Response: ! AA (cr) successful implement a calibration

Exception Response: ? AA (cr) command is entered incorrectly

Explain: ! Normal response delimiter; ? Abnormal response delimiter; (cr) carriage return character

Examples: $ 0110 (cr)

Module response:! 01 (cr)

Explain: The command $ 0110 (cr) is the offset calibration of channel 0 with module address 01H; $ is the delimiter; 01H is the module address; 1 is the keyword; 0 is the channel number; (cr) is a carriage return character

1.2.6 Read Channel Status Command

$ AA6

Name: Read channel status command

Syntax: $ AA6 (cr)

$ Delimiter

AA module address range is 00H to FFH

6 command keyword, here must be 6, otherwise other operations will be performed

(cr) is the terminating character, carriage return (0Dh)

Normal Response: ! AAxx (cr)

Abnormal Response: ? AA (cr) command is entered incorrectly

Explain:! Is the delimiter, which means normal response;? Is the delimiter, which means abnormal response; xx represents the channel state, if the bit is1, means the channel open, if is 0 means channel close; (cr) is a carriage return character

Example:

$ 016 (cr)

Module response:! 0105 (cr)

Explain: Command $ 016 (cr) is read address 01H module channel status; $ is the delimiter; 01H is the address; 6 is the keyword. Response message: ! Is the delimiter, means normal response; 01H is the module address; 05H (0000 0101B) indicates that the module's channel status is: 0 channels and 2 channels open, other channels are closed; (cr) terminating character with carriage value 0DH

1.2.7 Read Measuring Rang Command

$ AA8

Name: Read Measuring Rang Command

Syntax: $ AA8 (cr)

$ Delimiter

AA module address, range is from 00H to FFH

8 command keyword, here must be 8, or may perform other operations

(cr) is the terminating character, carriage return (0Dh)

Normal Response: ! AAxx (cr)

Abnormal Response:? AA (cr) command is entered incorrectly

Explain: ! Is the delimiter, means normal response;? Is the delimiter, means abnormal response; xx is the range of code, the code corresponding to the ranges shown in Table 1.2; (cr) terminating character with carriage value 0DH

Example:

$ 018 (cr)

Module responds:! 0105 (cr)

Explain: Command $ 018 (cr) is read address 01H module range; $ is the delimiter; 01H is the address; 8 keywords. Response message: ! Is the delimiter, means normal response; 01H is the module address;

05H indicates that the module's range is: ± 500mV; (cr) end of the transport operator whose value 0DH

Range Code	Measuring Range
00	±20mA
01	±1V
02	±2.5V
03	±5V
04	±10V
05	±200mV
06	±100mV
07	±75mV
08	±55mV
09	User-define

Form 1.2 Range code comparison table

1.2.8 Close or open channel command

$ AA5xx

Name: Open or close the channel command

Syntax: $ AA5xx (cr)

$ Delimiter

AA module address, range is from 00H to FFH

5 command keyword, here must be 5 , otherwise may perform other operations

xx Channel Status Code

Normal Response:! AA (cr)

Exception Response:? AA (cr) command input error

Explain:! Normal response delimiter;? Abnormal response delimiter; (cr) carriage return character

Example 1:

$ 0150A (cr)

Module responds:! 01 (cr)

Explain: Command $ 0150A (cr) is the channel status is 0AH (0000 1010B) with configuration address 01H,

means the channel 3 and channel 1 opens, the other closed; $ is the delimiter; 01H is the module address; 5 is a keyword ; 0AH channel status code; (cr) is the terminating character, carriage return (0Dh)

1.2.9 Read Module Name Command

$ AAM

Name: Read module name command

Syntax: $ AAM (cr)

$ Delimiter

AA module address, range is from 00H to FFH

M command keyword, here must be M, otherwise may perform other operations

(Cr) is the terminating character, carriage return (0Dh)

Normal Response:! AA (cr)

Exception Response:? AA (cr) command is entered incorrectly

Explain: ! delimiter, means normal response;? Delimiter: means abnormal response; (cr) carriage return terminator, its value is 0DH

Example:

$ 01M (cr)

Module Response:! 01DAM8804 (cr)

Explain: Command $ 01M (cr) is read address 01H module name, its read name is DAM8804; $ is the delimiter; 01H is the module address; M is the keyword. Response message:! Is the delimiter, means normal response; 01H is the module address; DAM8804 is the module name

1.2.10 Read firmware version command

$ AAV

Name: Read firmware version command

Syntax: $ AAV (cr)

$ Delimiter

AA module address, range is from 00H to FFH

V command keyword, here must be V, otherwise may perform other operations

(Cr) carriage return terminator, its value is 0DH

Normal Response:! AA (cr)

Exception Response:? AA (cr) command is entered incorrectly

Description: ! Is the delimiter, means normal response;? Is the delimiter, means abnormal response; (cr) carriage return terminator, its value is 0DH

Example:

$ 01V (cr)

Module Response:! 01Vr0.1 (cr)

Explain: Command $ 01M (cr) is read address 01H module firmware version, which read version is Vr0.1; $ is the delimiter; 01H is the module address; V is the keyword. Response message:! Is the delimiter, means normal response; 01H is the module address; Vr0.1 is the firmware version

1.2.11 Set the communication protocol command

$ AAPx

Name: Set the communication protocol command

Syntax: $ AAPx (cr)

$	Delimiter
AA	module address
P	command keyword, here must be a P, otherwise may perform other operations
X	one Binary, 0: ASCII protocol; 1: Indicates the MODBUS RTU protocol
(Cr)	carriage return terminator, its value is 0DH

Normal Response:! AA (cr)

Exception Response:? AA (cr) command input error or is not the configuration mode

Explain: The command must be executed under the configuration mode, other modes is invalid command will return the response. ! Normal response delimiter;? Abnormal response delimiter; (cr) carriage return character

Example:

$ 01P1 (cr)

Module Response:! 01 (cr)

Explain: Command $ 01P1 (cr) is the module address 01H communication protocol is MODBUS RTU protocol; $ is the delimiter; 01H is the address; P is the keyword; 1 means set the communication protocol is MODBUS RTU protocol.

1.2.12 set span command

$ AA7xx

Syntax: $ AA7xx

$ Delimiter

AA module address

7 command keyword, here must be 7, or may perform other operations

xx two hexadecimal number that represents the corresponding range, range code table shown in For -m 1.2

(Cr) carriage return terminator, its value is 0DH

Normal Response:! AA (cr)

Exception Response:? AA (cr) command input error or is not the configuration mode

Explain: This command only multi-range module (DAM-9xxx) support, and must be in configuration mode, other modes are invalid command and will return the response. ! Normal response delimiter;? Abnormal response delimiter; (cr) carriage return character

Example:

$ 12703 (cr)

Module Response:! 01

Explain: Command $ 12703 (cr) is the range of configuration module at the address 12H; $ is the delimiter; 12 is the address; 7 is a keyword; 03 indicate the setting of the module range is ± 5V.

2

2. Modbus RTU communication protocol

2.1 Modbus Summarize

DAM series modules support standard MODBUS protocol, format data transmission by means of RTU frame on the bus. every 8 byte is divided into two 4 bit hexadecimal characters of its information. main advantage of this mode is under the same baud rate, whose higher density of characters transmitted than under the in ASCII mode, each message must be continuous transmission.

Each Byte mode of RTU mode:

System Code: 8-bit binary, hexadecimal 0-9, A-F

Data bits: 1 start bit

8 data bits, LSB first delivery

Odd / even parity 1 bit; no parity 0 bit

Stop bit 1 (with parity); Stop bit 2 (no parity)

With parity: 1 stop bit; no parity, 2 stop bits

Error check: cyclic redundancy check (CRC)

Note: DAM series modules by usual 8-N-1 data bits transmitted. That is 1 start bit, 8 data bits, 1 stop bit.

2.2 Query response cycle

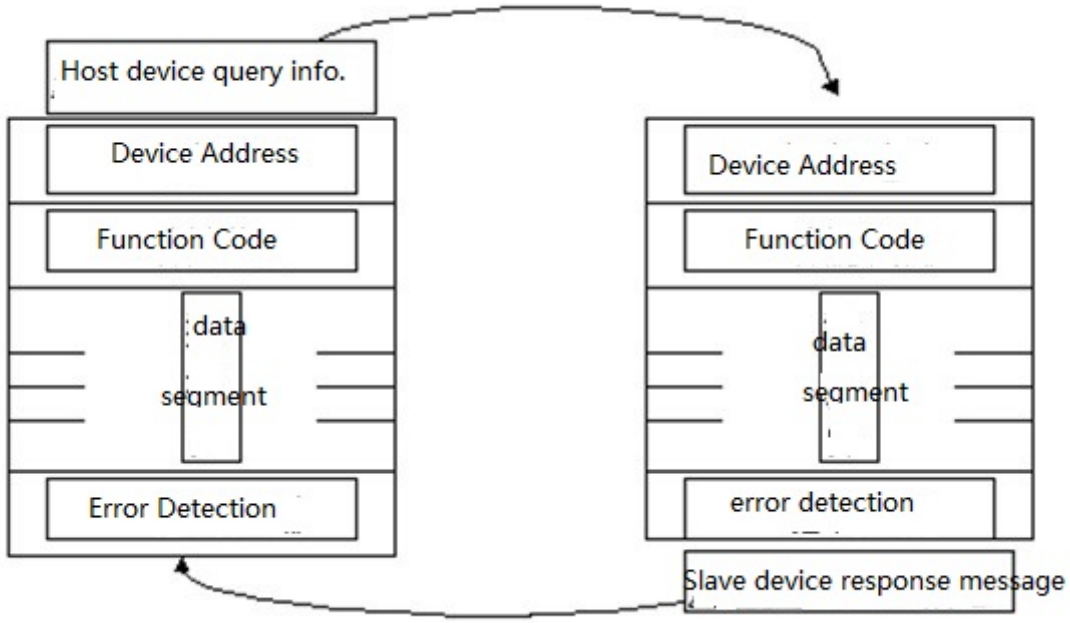

Figure 2.1 Host-Slave Query-response cycle

(1) Query

The function code in the query message can announce the selected slave device to perform what functions. Data segment contains any additional information that the slave device performs function. For example, function code 03 is required slave device to read keep registers and return their contents. Data segment must contain the information from the device advertised: which register start reading and the number of registers to be read. Error detection domain provides an authentication method for slave device to detect the message contents correct or not.

（2） Response

If the slave device generates a normal response, the function code in the response message is the response of query message function code. The data segment contains the data collected from the slave device: Like the register value or state. If an error occurs, the function code will be modified to respond to the message indicates errors, while the data segment contains a description of the error message code. Error detection domain allows the master device to confirm the message content is available or not.

2.3 Modbus Command

DAM module support 03/06 Modbus command. Command and responded format is as follow form. Supported Modbus address is 0x0000-0xffff. Once can read max 36 bytes.

2.3.1 Read Register command: 03

Host computer demand						
Address	Function code	First register high-order address	First register low order address	Register quantity high-order	Register quantity low-order	Error-checking
01	03	00	38	00	01	XX
Slave computer Response						
Address	Function code	Number of byte	Data high byte	Data low byte	Error-checking	
01	03	2	41	24	XX	
Hexadecimal number 4124 represents decimal integer 16676, error check value should be based on the transmission mode.						

2.3.1 Preset single register command:06

Host computer demand						
Address	Function code	Preset register’s high-order address	Preset register’s low-order address	Preset data high-order	Preset data low-order	Error-checking
01	06	00	01	00	0F	XX
Slave computer Response						
Address	Function code	Preset register’s high-order address	Preset register’s low-order address	Preset data high-order	Preset data low-order	Error-checking
01	06	00	01	00	0F	XX

2.3.2 DAM Module ModBus Address assignment Description:

Address	Data	Type	Description	Remark
40001	Module model	Read only	Module name, decided by module	ALL
40002	Channel status	Read/write	High-order read, low-order read and write	ALL
40003	Analog input 0	Read only	Channel 0 Data high 16 bit	ALL
40004	Analog input 1	Read only	Channel 1 Data high 16 bit	ALL
40005	Analog input 2	Read only	Channel 2 Data high 16 bit	Only 4/8/16 channel product
40006	Analog input 3	Read only	Channel 3 Data high 16 bit	Only 4/8/16 channel product
40007	Analog input 4	Read only	Channel 4 Data high 16 bit	Only 8/16 channel product
40008	Analog input 5	Read only	Channel 5 Data high 16 bit	Only 8/16 channel product
40009	Analog input 6	Read only	Channel6 Data high 16 bit	Only 8/16 channel product
40010	Analog input 7	Read only	Channel 7 Data high 16 bit	Only 8/16 channel product
40011	Analog input 8	Read only	Channel 8 Data high 16 bit	Only 16 channel product
40012	Analog input 9	Read only	Channel 9 Data high 16 bit	Only 16 channel product
40013	Analog input 10	Read only	Channel 10 Data high 16 bit	Only 16 channel product
40014	Analog input 11	Read only	Channel 11 Data high 16 bit	Only 16 channel product
40015	Analog input 12	Read only	Channel 12 Data high 16 bit	Only 16 channel product
40016	Analog input 13	Read only	Channel 13 Data high 16 bit	Only 16 channel product
40017	Analog input 14	Read only	Channel 14 Data high 16 bit	Only 16 channel product
40018	Analog input 15	Read only	Channel 15 Data high 16 bit	Only 16 channel product
40019	reserved	Read only	Read return 0	ALL
40020	reserved	Read only	Read return 0	ALL
……	…….	……	……	……
42000	Module Model	Read only	Module name, decided by module	ALL
42001	Channel Status	Read/write	High-order read, low-order read and write	ALL

42002	Analog input	Read only	Channel 0 data high 8 bit	ALL
42003	Analog input	Read only	Channel 0 data Low 16 bit	ALL
42004	Analog input	Read only	Channel 1 data high 8 bit	ALL
42005	Analog input	Read only	Channel 1 data Low 16 bit	ALL
42006	Analog input	Read only	Channel 2 data high 8 bit	Only 4/8/16 channel product
42007	Analog input	Read only	Channel 2 data Low 16 bit	Only 4/8/16 channel product
42008	Analog input	Read only	Channel 3 data high 8 bit	Only 4/8/16 channel product
42009	Analog input	Read only	Channel 3 data Low 16 bit	Only 4/8/16 channel product
42010	Analog input	Read only	Channel 4 data high 8 bit	Only 8/16 channel product
42011	Analog input	Read only	Channel 4 data Low 16 bit	Only 8/16 channel product
42012	Analog input	Read only	Channel 5 data high 8 bit	Only 8/16 channel product
42013	Analog input	Read only	Channel 5 data Low 16 bit	Only 8/16 channel product
42014	Analog input	Read only	Channel 6 data high 8 bit	Only 8/16 channel product
42015	Analog input	Read only	Channel 6 data Low 16 bit	Only 8/16 channel product
42016	Analog input	Read only	Channel 7 data high 8 bit	Only 8/16 channel product
42017	Analog input	Read only	Channel 7 data Low 16 bit	Only 8/16 channel product
42018	Analog input	Read only	Channel 8 data high 8 bit	Only 16 channel product
42019	Analog input	Read only	Channel 8 data Low 16 bit	Only 16 channel product
42020	Analog input	Read only	Channel 9 data high 8 bit	Only 16 channel product
42021	Analog input	Read only	Channel 9 data Low 16 bit	Only 16 channel product
42022	Analog input	Read only	Channel 10 data high 8 bit	Only 16 channel product
42023	Analog input	Read only	Channel 10 data Low 16 bit	Only 16 channel product
42024	Analog input	Read only	Channel 11 data high 8 bit	Only 16 channel product
42025	Analog input	Read only	Channel 11 data Low 16 bit	Only 16 channel product
42026	Analog input	Read only	Channel 12 data high 8 bit	Only 16 channel product
42027	Analog input	Read only	Channel 12 data Low 16 bit	Only 16 channel product
42028	Analog input	Read only	Channel 13 data high 8 bit	Only 16 channel product
42029	Analog input	Read only	Channel 13 data Low 16 bit	Only 16 channel product
42030	Analog input	Read only	Channel 14 data high 8 bit	Only 16 channel product

42031	Analog input	Read only	Channel 14 data Low 16 bit	Only 16 channel product
42032	Analog input	Read only	Channel 15 data high 8 bit	Only 16 channel product
42033	Analog input	Read only	Channel 15 data Low 16 bit	Only 16 channel product
42034	Reserved	Read only	Read is 0	ALL
…….				

Remark: DAM Series module each model storage address is different from channel, when read non storage data address, all will return to 0. Specific refer to upper form.

3

3. Disclaimer

Copyright

The text stated and related software in this manual are copyright of Amazing Electronic (HK) Limited, All property is absolute protection under national law, without the authorization, other companies, organizations, agencies and individuals shall not illegal to use and copy, otherwise it will be subject to national laws and sanctions.

The right to amend the text:

Amazing Electronic (HK) Limited reserves the right to amend the text without notification.

Appendix 1: Cycling Redundancy checking (CRC) C language origin code

/* CRC high-order byte value table */

flash uint8 arrCRCHi[256] =

{

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,

0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,

0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

```
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
/* CRC low order byte value table*/
flash    uint8    arrCRCLo[256] =
{
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04,
0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8,
0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10,
0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C,
0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0,
0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C,
0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54,
0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
```

```
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40
};
/*****************************************************************************/
/*Function name：uint16 crc16(uint8* puchMsg, uint8 usDataLen)                */
/* input parameter：non                                                       */
/* Response parameter: non                                                   */
/* function function：crc checking                                           */
/*****************************************************************************/
uint16 ModbusCrc16(uint8* datafp, uint8 datalen)
{
    uint8   u8CRCHi = 0xFF ;            /* high CRC byte initialize   */
    uint8   u8CRCLo = 0xFF ;            /* low CRC byte initialize */
    uint16 uIndex ;                     /* CRC index in circulation   */
    while (datalen--)                   /* transmission message buffer   */
    {
      uIndex   = u8CRCHi ^ * datafp++ ;   /*  计算 CRC          */
      u8CRCHi = u8CRCLo ^ arrCRCHi[uIndex] ;
      u8CRCLo = arrCRCLo[uIndex];
    }
    return (((uint16)u8CRCHi) << 8) + u8CRCLo;
}
```

Contact us◆

Amazing Electronic (HK) Limited
ADD: No. 520, Building 1, Jiamei Industrial Park,
Longwei Road, Futian District, Shenzhen,China. 518049.
Email:info@amazingelectronic.com