AIO Library

內容

[1 Install library 2](#_Toc388448966)

[1.1 X86: 2](#_Toc388448967)

[1.2 Matrix tool chain arm-linux-gnueabi-gcc: 2](#_Toc388448968)

[1.3 Matrix tool chain arm-linux- gcc: 2](#_Toc388448969)

[2 Make example 2](#_Toc388448970)

[2.1 X86: 2](#_Toc388448971)

[2.2 Matrix arm-linux-gnueabi-gcc: 2](#_Toc388448972)

[2.3 Matrix arm-linux- gcc: 2](#_Toc388448973)

[3 API Reference 2](#_Toc388448974)

[3.1 IO 2](#_Toc388448975)

[3.1.1 AIO\_Init 2](#_Toc388448976)

[4.1.1 AIO\_Connect 2](#_Toc388448977)

[4.1.2 AIO\_Close 3](#_Toc388448978)

[4.1.3 AIO\_RIO\_2018\_DO\_Read 3](#_Toc388448979)

[4.1.4 AIO\_RIO\_2018\_DO\_Write 3](#_Toc388448980)

[4.1.5 AIO\_RIO\_2018\_DI\_Read 4](#_Toc388448981)

[4.1.6 AIO\_RIO\_2018\_TC\_Read 4](#_Toc388448982)

[4.1.7 AIO\_RIO\_2017\_DO\_Read 5](#_Toc388448983)

[4.1.8 AIO\_RIO\_2017\_DO\_Write 5](#_Toc388448984)

[4.1.9 AIO\_RIO\_2017\_AI\_Read 5](#_Toc388448985)

[4.1.10 AIO\_RIO\_2010\_DO\_Read 6](#_Toc388448986)

[4.1.11 AIO\_RIO\_2010\_DO\_Write 6](#_Toc388448987)

[4.1.12 AIO\_RIO\_2010\_DO\_Write\_bit 7](#_Toc388448988)

[4.1.13 AIO\_RIO\_2010\_DI\_Read 7](#_Toc388448989)

[4.1.14 AIO\_RIO\_2010\_Sensor\_Read 8](#_Toc388448990)

[4.2 Configure 8](#_Toc388448991)

[4.2.1 AIO\_Cfg\_Login 8](#_Toc388448992)

[4.2.2 AIO\_Cfg\_RIO\_2018\_Get\_TC\_Enable 9](#_Toc388448993)

[4.2.3 AIO\_Cfg\_RIO\_2018\_Set\_TC\_Enable 9](#_Toc388448994)

[4.2.4 AIO\_Cfg\_RIO\_2017\_Get\_AI\_Enable 10](#_Toc388448995)

[4.2.5 AIO\_Cfg\_RIO\_2017\_Set\_AI\_Enable 10](#_Toc388448996)

[4.2.6 AIO\_Cfg\_RIO\_2017\_Get\_AI\_Range 11](#_Toc388448997)

[4.2.7 AIO\_Cfg\_RIO\_2017\_Set\_AI\_Range 11](#_Toc388448998)

[4.2.8 AIO\_Cfg\_SaveExit 12](#_Toc388448999)

[4.2.9 AIO\_Cfg\_Exit 12](#_Toc388449000)

1. Install library
   1. X86:
      1. copy x86\_gcc.tar到/底下
      2. tar –xvf x86\_gcc.tar
      3. libaio\_x86.so會被放在/usr/local/lib中
   2. Matrix tool chain arm-linux-gnueabi-gcc:
      1. copy arm\_eabigcc.tar到/底下
      2. tar –xvf arm\_eabigcc.tar
      3. libaio\_9G20.so會被放在/usr/local/arm-linux-gnueabi/lib/中
   3. Matrix tool chain arm-linux- gcc:
      1. 無需安裝，libaio\_9200.so被放在Example中
2. Make example
   1. X86:
      1. make –f makefile\_gcc
      2. 執行檔案會被放在./output/gcc中
   2. Matrix arm-linux-gnueabi-gcc:
      1. make –f makefile\_eabigcc
      2. 執行檔案會被放在./output/eabigcc中
   3. Matrix arm-linux- gcc:
      1. make –f makefile\_armgcc
      2. 執行檔案會被放在./output/armgcc中
      3. libaio\_9200.so在./output/armgcc/armgcc\_lib中
3. API Reference
   1. IO
      1. AIO\_Init
         1. int AIO\_Init(void)
4. Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to AIO.h |

* + 1. AIO\_Connect
       1. int AIO\_Connect(char \*ip\_str, int port, int tmout\_ms, int \*socket\_fd)

Arguments:

|  |  |
| --- | --- |
| char \*ip\_str [in] | Device IP address |
| int port [in] | Modbus TCP listen port |
| int tmout\_ms [in] | connect response timeout. The unit is in milliseconds. |
| int \*socket\_fd [out] | Socket file handle of the connection |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to AIO.h |

* + 1. AIO\_Close
       1. int AIO\_Close(int socket\_fd)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to AIO.h |

* + 1. AIO\_RIO\_2018\_DO\_Read
       1. int AIO\_RIO\_2018\_DO\_Read(int socket\_fd, int tmout\_ms, unsigned char \*value)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| unsigned char \*value [out] | stores the DO value  only one DO for RIO-2018 |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2018\_AIO.h |

* + 1. AIO\_RIO\_2018\_DO\_Write
       1. int AIO\_RIO\_2018\_DO\_Write(int socket\_fd, int tmout\_ms, unsigned char value)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| unsigned char value [in] | stores the DO value |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2018\_AIO.h |

* + 1. AIO\_RIO\_2018\_DI\_Read
       1. int AIO\_RIO\_2018\_DI\_Read(int socket\_fd, int tmout\_ms, unsigned char start\_no, unsigned char quantity, unsigned char \*value\_buf, int buf\_size, int \* value\_byte\_count)
       2. Response data example
          1. value\_buf[0]:

bit0- the first DI, bit1- the second DI

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_no [in] | the number of the first DI. DI#1: 1 |
| int quantity [in] | quantity of DI(The maximum number is 2) |
| unsigned char \* value\_buf [out] | an array that stores the DI value. |
| int buf\_size [in] | size of the value \_buf |
| int \* value\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2018\_AIO.h |

* + 1. AIO\_RIO\_2018\_TC\_Read
       1. int AIO\_RIO\_2018\_TC\_Read(int socket\_fd, int tmout\_ms, unsigned char start\_channel, unsigned char quantity,unsigned char \*value\_buf, int buf\_size, int \* value\_byte\_count)
       2. AI response see “RIO Modbus Function List” and aio example

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_ channel [in] | the channel of the first TC，TC#1: 1 |
| int quantity [in] | quantity of TC(The maximum number is 3) |
| unsigned char \*value\_buf [out] | an array that stores the TC value |
| int buf\_size [in] | size of the response\_buf |
| int \* value\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2018\_AIO.h |

* + 1. AIO\_RIO\_2017\_DO\_Read
       1. int AIO\_RIO\_2017\_DO\_Read(int socket\_fd, int tmout\_ms, unsigned char \*value)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| unsigned char \*value [out] | stores the DO value  only one DO for RIO-2017 |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_RIO\_2017\_DO\_Write
       1. int AIO\_RIO\_2017\_DO\_Write(int socket\_fd, int tmout\_ms, unsigned char value)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| unsigned char value [in] | stores the DO value |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_RIO\_2017\_AI\_Read
       1. int AIO\_RIO\_2017\_AI\_Read(int socket\_fd, int tmout\_ms, unsigned char start\_channel, unsigned char quantity,unsigned char \*value\_buf, int buf\_size, int \* value\_byte\_count)
       2. AI response see “RIO Modbus Function List” and aio example

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_ channel [in] | the channel of the first AI，AI#1: 1 |
| int quantity [in] | quantity of AI(The maximum number is 8) |
| unsigned char \*value\_buf [out] | an array that stores the AI value |
| int buf\_size [in] | size of the response\_buf |
| int \* value\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_RIO\_2010\_DO\_Read
       1. int AIO\_RIO\_2010\_DO\_Read(int socket\_fd, int tmout\_ms, unsigned char start\_no, unsigned char quantity, unsigned char \* value)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_no [in] | the number of the first DO. DO#1: 1 |
| int quantity [in] | quantity of DO(The maximum number is 8) |
| unsigned char \*value [out] | stores the DO value  bit0-first DO… |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2010\_AIO.h |

* + 1. AIO\_RIO\_2010\_DO\_Write
       1. int AIO\_RIO\_2010\_DO\_Write(int socket\_fd, int tmout\_ms, unsigned char start\_no, unsigned char quantity, unsigned char value)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_no [in] | the number of the first DO. DO#1: 1 |
| int quantity [in] | quantity of DO(The maximum number is 8) |
| unsigned char value [in] | stores the DO value  bit0-first DO… |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2010\_AIO.h |

* + 1. AIO\_RIO\_2010\_DO\_Write\_bit
       1. int AIO\_RIO\_2010\_DO\_Write\_bit(int socket\_fd, int tmout\_ms, unsigned char start\_no, unsigned char value)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_no [in] | the number of the first DO. DO#1: 1 |
| unsigned char value [in] | bit0: stores the DO value |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2010\_AIO.h |

* + 1. AIO\_RIO\_2010\_DI\_Read
       1. int AIO\_RIO\_2010\_DI\_Read(int socket\_fd, int tmout\_ms, unsigned char start\_no, unsigned char quantity, unsigned char \*value\_buf, int buf\_size, int \* value\_byte\_count)
       2. Response data example
          1. value\_buf[0]:

bit0- the first DI, bit7- the eighth DI

* + - * 1. value\_buf[1]:

bit0- the 9th DI, bit7-the 16th DI

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_no [in] | the number of the first DI. DI#1: 1 |
| int quantity [in] | quantity of DI(The maximum number is 16) |
| unsigned char \* value\_buf [out] | an array that stores the DI value. |
| int buf\_size [in] | size of the value \_buf |
| int \* value\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2010\_AIO.h |

* + 1. AIO\_RIO\_2010\_Sensor\_Read
       1. int AIO\_RIO\_2010\_Sensor\_Read(int socket\_fd, int tmout\_ms, unsigned char \*value\_buf, int buf\_size, int \*response\_byte\_count)
       2. AI response see “RIO Modbus Function List” and aio example

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for modbus connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| unsigned char \*value\_buf [out] | an array that stores the sensor value |
| int buf\_size [in] | size of the response\_buf |
| int \*response\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2010\_AIO.h |

* 1. Configure
     1. AIO\_Cfg\_Login
        1. int AIO\_Cfg\_Login(char \*ip\_str, int port, int tmout\_ms, char \*password, int \*socket\_fd)

Arguments:

|  |  |
| --- | --- |
| char \*ip\_str [in] | Device IP address |
| int port [in] | Command mode TCP listen port |
| int tmout\_ms [in] | connect response timeout. The unit is in milliseconds. |
| char \*password | Command mode password. |
| int \*socket\_fd [out] | Socket file handle of the connection |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_Cfg\_RIO\_2018\_Get\_TC\_Enable
       1. int AIO\_Cfg\_RIO\_2018\_Get\_TC\_Enable(int socket\_fd, int tmout\_ms, int start\_channel, int quantity, unsigned char \*response\_buf, int buf\_size, int \*response\_byte\_count)
       2. Response data example
          1. response\_buf[0]:

bit0- the first TC, bit2- the third TC

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_channel [in] | the channel of the first TC. TC#1: 1 |
| int quantity [in] | quantity of TC |
| unsigned char \*response\_buf [out] | an array that stores the TC enable value or error response |
| int buf\_size [in] | size of the response\_buf |
| int \*response\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2018\_AIO.h |

* + 1. AIO\_Cfg\_RIO\_2018\_Set\_TC\_Enable
       1. int AIO\_Cfg\_RIO\_2018\_Set\_TC\_Enable(int socket\_fd, int tmout\_ms, int start\_channel, int quantity, int set\_value)
       2. set value example
          1. bit0- the first TC, bit7- the third TC

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_channel [in] | the channel of the first TC. TC#1: 1 |
| int quantity [in] | quantity of TC |
| int set\_value | AI enable value |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2018\_AIO.h |

* + 1. AIO\_Cfg\_RIO\_2017\_Get\_AI\_Enable
       1. int AIO\_Cfg\_RIO\_2017\_Get\_AI\_Enable(int socket\_fd, int tmout\_ms, int start\_channel, int quantity, unsigned char \*response\_buf, int buf\_size, int \*response\_byte\_count)
       2. Response data example
          1. response\_buf[0]:

bit0- the first AI, bit7- the eighth AI

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_channel [in] | the channel of the first AI. AI#1: 1 |
| int quantity [in] | quantity of AI |
| unsigned char \*response\_buf [out] | an array that stores the AI enable value or error response |
| int buf\_size [in] | size of the response\_buf |
| int \*response\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_Cfg\_RIO\_2017\_Set\_AI\_Enable
       1. int AIO\_Cfg\_RIO\_2017\_Set\_AI\_Enable(int socket\_fd, int tmout\_ms, int start\_channel, int quantity, int set\_value)
       2. set value example
          1. bit0- the first AI, bit7- the eighth AI

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_channel [in] | the channel of the first AI. AI#1: 1 |
| int quantity [in] | quantity of AI |
| int set\_value | AI enable value |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_Cfg\_RIO\_2017\_Get\_AI\_Range
       1. int AIO\_Cfg\_RIO\_2017\_Get\_AI\_Range(int socket\_fd, int tmout\_ms, int start\_channel, int quantity, unsigned char \*response\_buf, int buf\_size, int \*response\_byte\_count)
       2. Response data example
          1. response\_buf[0]: the first AI
          2. response\_buf[1]: the second AI

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_ channel [in] | the channel of the first AI. AI#1: 1 |
| int quantity [in] | quantity of AI |
| unsigned char \*response\_buf [out] | an array that stores the AI range value or error response |
| int buf\_size [in] | size of the response\_buf |
| int \*response\_byte\_count [out] | the response data length |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_Cfg\_RIO\_2017\_Set\_AI\_Range
       1. int AIO\_Cfg\_RIO\_2017\_Set\_AI\_Range(int socket\_fd, int tmout\_ms, int start\_channel, int quantity, unsigned char \*request\_buf, int buf\_size)
       2. request data example
          1. request\_buf[0]: the first AI
          2. request\_buf[1]: the second AI

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |
| int tmout\_ms [in] | read response timeout. The unit is in milliseconds. |
| int start\_channel [in] | the channel of the first AI. AI#1: 1 |
| int quantity [in] | quantity of AI |
| unsigned char \*request\_buf [out] | an array that stores the AI range value or error request |
| int buf\_size [in] | size of the request\_buf |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_Cfg\_SaveExit
       - 1. int AIO\_Cfg\_SaveExit(int socket\_fd):

save the changes and reboot

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |

* + 1. AIO\_Cfg\_Exit
       1. int AIO\_Cfg\_Exit(int socket\_fd)

Arguments:

|  |  |
| --- | --- |
| int socket\_fd [in] | file handle for command mode connection |

Return Value:

|  |  |
| --- | --- |
| Succeed | AIO\_OK |
| Fail | Refer to RIO\_2017\_AIO.h |