RIO-2015 API Reference

2016-03-18

目錄

[1 Ethernet 1](#_Toc446085125)

[2 RTC(Real-time Clock) 2](#_Toc446085126)

[3 DIO 3](#_Toc446085127)

[4 Web 3](#_Toc446085128)

[5 Serial USART 4](#_Toc446085129)

[6 Analog Output 6](#_Toc446085130)

[7 SPI Flash(2M bytes) 6](#_Toc446085131)

[8 SD card 7](#_Toc446085132)

[9 EEPROM(2K bytes) 7](#_Toc446085133)

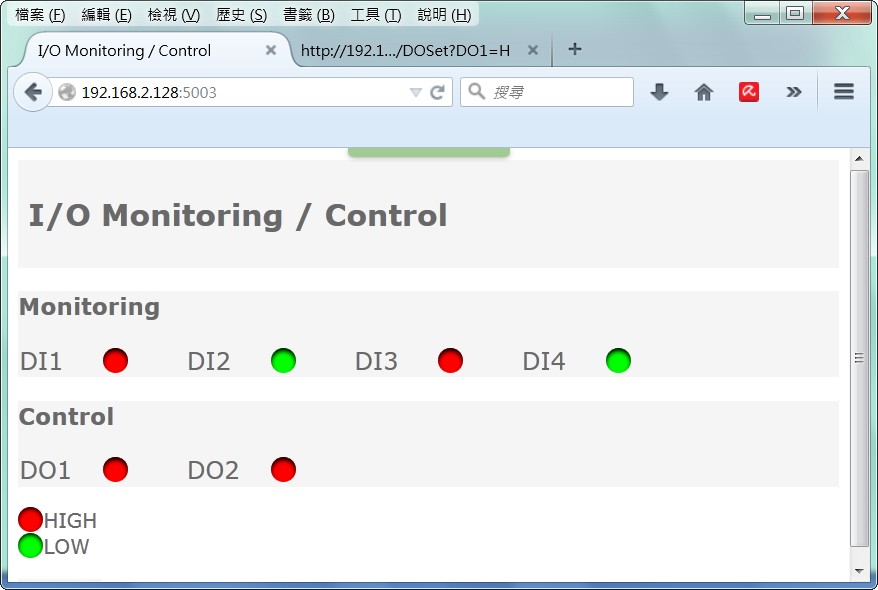
[10 SPI SRAM(128K bytes) 7](#_Toc446085134)

[11 Analog Input 8](#_Toc446085135)

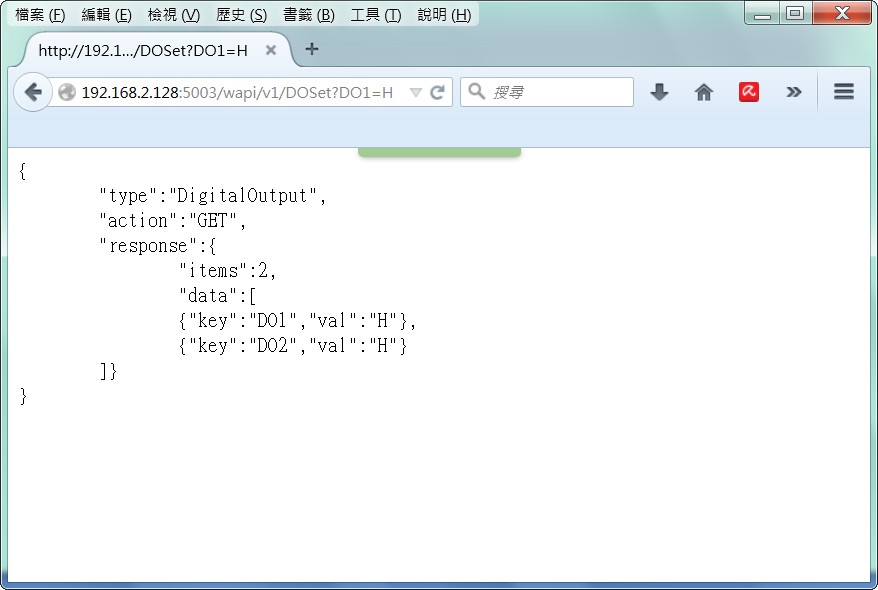
[12 One Wire Sensor 8](#_Toc446085136)

[13 Float to string 8](#_Toc446085137)

1. Ethernet
   1. Example: E01\_TCPServer
      1. netconn\_new
         1. struct netconn \* netconn\_new(enum netconn type type)
      2. netconn\_bind
         1. err\_t netconn\_bind(struct netconn \*conn, ip\_addr\_t \*addr, u16\_t port)
      3. netconn\_listen
         1. err\_t netconn\_ listen(struct netconn \*conn)
      4. netconn\_accept
         1. err\_t netconn\_accept(struct netconn \*conn, struct netconn \*\*new\_conn)
      5. netconn\_recv
         1. err\_t netconn\_recv(struct netconn \*conn, struct netbuf \*\*new\_buf)
      6. netconn\_write
         1. err\_t netconn\_write(struct netconn \*conn, const void \*dataptr, size\_t size, u8\_t apiflags)
      7. netconn\_close
         1. err\_t netconn\_close(struct netconn \*conn)
         2. Closes the connection conn
      8. netconn\_delete
         1. err\_t netconn\_delete(struct netconn \*conn)
         2. Deallocates the netconn conn. If the connection is open, it is closed as a result of this call
      9. netbuf\_data
         1. err\_t netbuf\_data (struct netbuf \*buf, void \*\*dataptr, u16\_t \*len)
      10. netbuf\_next
          1. s8\_t netbuf\_next (struct netbuf \*buf)
      11. netbuf\_delete
          1. void netbuf\_delete (struct netbuf \*buf)
   2. Example: E60\_MQTT
      1. BSD socket library
         1. int socket(int domain, int type, int protocol)
         2. int bind(int s, const struct sockaddr \*name, socklen\_t namelen)
         3. int listen(int s, int backlog)
         4. int accept(int s, struct sockaddr \*addr, socklen\_t \*addrlen)
         5. int recv(int s, void \*mem, size\_t len, int flags)
         6. int send(int s, void \*dataptr, size\_t size, int flags)
         7. int closesokcet(int s)
2. RTC(Real-time Clock)
   1. Example: E02\_RTC
   2. set\_rtc\_time
      1. uint8\_t set\_rtc\_time(uint8\_t u\_hour, uint8\_t u\_minute, uint8\_t u\_second)
   3. set\_rtc\_date
      1. uint8\_t set\_rtc\_date(uint16\_t us\_year, uint8\_t u\_month, uint8\_t u\_day, uint8\_t u\_week)
   4. get\_rtc\_time
      1. void get\_rtc\_time(uint8\_t \*pu\_hour, uint8\_t \*pu\_minute, uint8\_t \*pu\_second)
   5. get\_rtc\_date
      1. void get\_rtc\_date(uint16\_t \*pus\_year, uint8\_t \*pu\_month, uint8\_t \*pu\_day, uint8\_t \*pu\_week)
   6. error define: RIO2015\_RTC.h
   7. 待完成項目
      1. 自動校時Enable/Disable，自動校時發生時間
      2. time zone
3. DIO
   1. Example: E03\_DIO
   2. set\_do\_value
      1. uint8\_t set\_do\_value(uint8\_t do\_no, uint8\_t do\_val)
      2. do\_no: 1~2
      3. do\_val: 0 or 1
   3. get\_do\_value
      1. uint8\_t get\_do\_value(uint8\_t do\_no)
   4. get\_do\_port\_value
      1. uint8\_t get\_do\_port\_value(void)
      2. bit0: DO#1, bit1: DO#2
   5. get\_di\_value
      1. uint8\_t get\_di\_value(uint8\_t di\_no)
      2. di\_no: 1~4
   6. get\_di\_port\_value
      1. uint8\_t get\_di\_port\_value(void)
      2. bit0: DI#1, bit1: DI#2, bit2: DI#3, bit3: DI#4
4. Web
   1. Example: E04\_Web\_DIO
   2. Web File
      1. html等client端的檔案，存放在0x004A0000開始的flash中
      2. 用Manager Utility把檔案轉換打包成.anf檔案，再upgrade到device中, 如Examples\E04\_Web\_DIO\WEB中的mnt\_ctrl.anf
   3. Web API
      1. 如同AJAX命令，可以由程式動態處理
      2. void user\_web\_init(unsigned short (\* user\_web)(char \*name, char \*arg, char \*ptr\_data\_buf));
         1. 註冊user\_web function call
         2. user\_web取得命令並回應，如web\_ajax.c的web\_file\_open
      3. index.html畫面如下圖：



* + 1. 在地址欄設定DO值，如下圖：



1. Serial USART
   1. Example: E05\_SerialUSART
   2. serial\_usart\_open
      1. uint8\_t serial\_usart\_open(usart\_if p\_usart, uint32\_t baudrate, uint32\_t parity\_type, uint32\_t char\_length, uint32\_t stop\_bits)
      2. p\_usart: USART0 : RS-485, USART1: RS-232
   3. serial\_usart\_set\_read\_timeout
      1. uint8\_t serial\_usart\_set\_read\_timeout(usart\_if p\_usart, uint32\_t timeout\_ms)
      2. p\_usart: USART0 : RS-485, USART1: RS-232
      3. timeout\_ms:
         1. =0: serial\_usart\_read will block, 直到收到data長度等於serial\_usart\_read中的wanted\_len
         2. =1: serial\_usart\_read即刻返回當前收到data的長度(default)
         3. >1:
            1. 若當前data長度<wanted\_len，等待timeout\_ms毫秒後，serial\_usart\_read返回當前收到data的長度。
            2. 若當前data長度>=wanted\_len，serial\_usart\_read即刻返回wanted\_len
   4. serial\_usart\_set\_write\_timeout
      1. uint8\_t serial\_usart\_set\_write\_timeout(usart\_if p\_usart, uint32\_t timeout\_ms)
      2. p\_usart: USART0 : RS-485, USART1: RS-232
      3. timeout\_ms:
         1. =0: block, 直到serial\_usart\_write的data都發送完畢
         2. >0: (default=1)
            1. 若timeout\_ms毫秒內，serial\_usart\_write的data沒有發送完畢，即刻返回已發送的長度
   5. serial\_usart\_read
      1. uint16\_t serial\_usart\_read(usart\_if p\_usart, uint8\_t \*rbuf, uint16\_t wanted\_len)
      2. 依serial\_usart\_set\_read\_timeout設定執行
         1. =0: block, 直到收到data長度等於wanted\_len
         2. =1:即刻返回當前收到data的長度
         3. >1:
            1. 若當前data長度<wanted\_len，等待read\_timeout毫秒後，返回當前收到data的長度。
            2. 若當前data長度>=wanted\_len，即刻返回wanted\_len
   6. serial\_usart\_write
      1. uint16\_t serial\_usart\_write(usart\_if p\_usart, uint8\_t \*wbuf, uint16\_t write\_len)
      2. 依serial\_usart\_set\_write\_timeout設定執行
         1. =0: block, 直到data都發送完畢
         2. >0: (default=1)
            1. 若write\_timeout毫秒內， data沒有發送完畢，即刻返回已發送的長度
2. Analog Output
   1. Example: E06\_AO
      1. ao\_init
         1. uint8\_t ao\_init(void)
      2. ao\_channel\_init
         1. uint8\_t ao\_channel\_init(uint8\_t enable, uint8\_t channel\_no, uint8\_t range, uint16\_t gain, uint16\_t offset)
         2. gain: 0xFFFF
         3. offset: 0x8000
      3. ao\_channel\_set\_voltage
         1. uint8\_t ao\_channel\_set\_voltage(uint8\_t channel\_no, float set\_voltage, float \*real\_voltage)
      4. ao\_channel\_set\_current
         1. uint8\_t ao\_channel\_set\_current(uint8\_t channel\_no, float set\_current, float \*real\_current)
      5. ao\_channel\_get\_status
         1. uint8\_t ao\_channel\_get\_status(void)
      6. ao\_channel\_get\_voltage
         1. uint8\_t ao\_channel\_get\_voltage(uint8\_t channel\_no, uint8\_t \*range, float \*set\_voltage, float \*real\_voltage)
      7. ao\_channel\_get\_current
         1. uint8\_t ao\_channel\_get\_current(uint8\_t channel\_no, uint8\_t \*range, float \*set\_voltage, float \*real\_voltage)
      8. range
         1. 0 V to 5 V voltage range
         2. 0 V to 10 V voltage range
         3. -5 V to +5 V voltage range
         4. -10 V to 10 V voltage range
         5. 4 mA to 20 mA current range
         6. 0 mA to 20 mA current range
         7. 0 mA to 24 mA current range
3. SPI Flash(2M bytes)
   1. Example: E07\_SPI\_FLASH
      1. spi\_flash\_init
         1. uint8\_t spi\_flash\_init(void)
      2. spi\_flash\_erase\_chip
         1. uint8\_t spi\_flash\_erase\_chip(void)
      3. spi\_flash\_erase\_sector
         1. uint8\_t spi\_flash\_erase\_sector(uint16\_t sector\_no)
         2. sector no: 0~511, sector size: 4Kbytes
      4. spi\_flash\_erase\_block
         1. uint8\_t spi\_flash\_erase\_block(uint16\_t block\_no)
         2. block no: 0~31, block size: 64Kbytes
      5. spi\_flash\_write\_page
         1. uint8\_t spi\_flash\_write\_page(uint16\_t page\_no, uint8\_t \*data\_buf, uint16\_t len)
         2. page no: 0~8191, page size: 256bytes
      6. spi\_flash\_read
         1. uint8\_t spi\_flash\_read(uint32\_t start\_address, uint8\_t \*data\_buf, uint32\_t len)
4. SD card
   1. Example: E08\_SDcard
      1. is\_sd\_present
         1. uint8\_t is\_sd\_present(void)
         2. if mount ok, return 1
      2. see RIO2015\_SD.h
5. EEPROM(2K bytes)
   1. Example: E09\_EEPROM
      1. rio\_eeprom\_write\_byte
         1. uint8\_t rio\_eeprom\_write\_byte(uint16\_t address, uint8\_t data\_byte)
         2. write one byte to eeprom
      2. rio\_eeprom\_write\_page
         1. uint8\_t rio\_eeprom\_write\_page(uint16\_t address, uint8\_t \*data\_buf, uint16\_t len)
         2. write full page to eeprom, “len” should be 16, “address” should be multiple of 16
      3. rio\_eeprom\_read
         1. uint8\_t rio\_eeprom\_read(uint16\_t address, uint8\_t \*data\_buf, uint16\_t len)
6. SPI SRAM(128K bytes)
   1. Example: E07\_ SPI\_SRAM
      1. spi\_sram\_init
         1. uint8\_t spi\_ sram\_init(void)
      2. spi\_sram\_write
         1. uint8\_t spi\_sram\_write(uint32\_t start\_address, uint8\_t \*data\_buf, uint16\_t len)
      3. spi\_sram\_read
         1. uint8\_t spi\_sram\_read(uint32\_t start\_address, uint8\_t \*data\_buf, uint32\_t len)
7. Analog Input
   1. Example: E11\_AI
   2. ai\_channel\_init
      1. uint8\_t ai\_channel\_init(uint8\_t enable, uint8\_t channel\_no, uint8\_t range)
      2. enable: 0x01=enable, 0x00=disable
      3. channel\_no: 1 ~ 4
      4. range:
         1. 0mV~150mV, -150mV~150mV
         2. 0mV~500mV, -500mV~500mV
         3. 0V~1V, -1V~1V
         4. 0V~5V, -5V~5V
         5. 0V~10V, -10V~10V
         6. 0~20mA
   3. ai\_channel\_get\_value
      1. uint8\_t ai\_channel\_get\_value(uint8\_t channel\_no, float \*value\_float)
      2. 此值為最近6個讀值的平均值
      3. 採樣頻率60Hz
8. One Wire Sensor
   1. Example: E12\_ OneWireSensor
   2. temperature\_sensor\_init
      1. uint8\_t temperature\_sensor\_init(void)
   3. temperature\_sensor\_get\_value
      1. uint8\_t temperature\_sensor\_get\_value(uint8\_t \*sensor\_id, float \*value\_float)
      2. 此值為最近3個讀值的平均值
      3. 背景執行1秒讀一次溫度值
9. Float to string
   1. E06\_AO, E11\_AI, E12\_OneWireSensor
   2. ftoa
      1. void ftoa(float n, char \*res, int afterpoint)
      2. 顯示float值。當用%f顯示float，在SRAM資源緊張的時候，會有問題。所以建議用ftoa列印float值