Multi IO v3

MÓDULO DE AUTOMAÇÃO IP

Manual de Integração Multi IO

Commbox

www.commbox.com.br

Sumário

Notas

Data	Versão	Autor	Revisor	FW	Conteúdo
03/2022	1.0	Fábio Pinheiro	Daniel Raupp		Criação do Manual
07/2022	1.1	David Santi			Detalhamento das portas de comunicação
08/2022	1.2	Daniel Raupp			Adicionado protocolo Commbox

1. Introdução

Este manual tem o objetivo de auxiliar no desenvolvimento de integrações com a Multi I/O v3. Estão disponíveis para integração os protocolos HTTP, BacNet e OnVif, além do protocolo Commbox.

A Multi I/O v3 comunica-se através da rede IP, e para cada protocolo é utilizado uma porta distinta, conforme tabela abaixo.

Protocolo	Porta
Commbox (UDP + TCP)	4091
HTTP	80
OnVif	8080
OnVif – WS Discovery (UDP)	3702
BacNet (UDP)	47808

Os detalhes para integração com cada um destes protocolos são encontrados nos capítulos seguintes.

2. Integração via protocolo HTTP

2.1 Introdução HTTP

Este capítulo tem como objetivo descrever as informações necessárias para que desenvolvedores de software realizem a integração com os produtos da série MIO V3 através dos protocolos HTPP.

Lista de Abreviaturas

HTTP	Hypertext Transfer Protocol
ID	Identificador
IP	Internet Protocol
JSON	JavaScript Object Notation
ms	Milésimos de segundos
PHP	Hypertext Preprocessor
S	Segundos

2.2 Protocolo HTTP

Com a integração é possível realizar a execução de comandos na MIO V3 através de requisições HTTP, que são geradas a partir de URLs endereçadas ao IP do equipamento.

Para facilitar a demonstração das requisições e dos comandos descritos neste documento, foram criados exemplos que utilizam um navegador de internet (Chrome, Firefox, Internet Explorer e etc) para enviar as requisições e receber as respostas da MIO. Nesses exemplos, as requisições HTTP são originadas através do navegador onde, na sua barra de endereços, é informada a URL e os parâmetros para o acionamento de comandos na MIO. As respostas dos comandos são exibidas no corpo da página do navegador ao término da requisição.

Também são demonstrados exemplos de requisições HTTP utilizando a linguagem de programação PHP. Vale ressaltar que a integração pode ser realizada em qualquer linguagem de programação.

Este documento limita-se em descrever informações relacionadas a integração com a MIO V3. Portanto, não serão explicados conceitos sobre o protocolo HTTP e linguagens de programação.

Este documento é válido para os produtos da série MIO V3 com versão de firmware 3.02 ou superior.

2.3 Definições Gerais

Quantidade de entradas e saídas:

A série MIO possui 5 modelos de placas, sendo que cada modelo possui uma quantidade específica de entradas e saídas, conforme mostrado na tabela 1:

Modelo MIO	Quantidade de entradas	Quantidade de saídas (relés)
MIO402	4	2
MIO400	4	4
MIO800	8	8
MIO2408	24	8
MIO0816	8	16

Tabela 1 – Modelo MIO x Quantidade de entradas e relés.

Em requisições que precisam indicar qual entrada ou relé deseja-se utilizar, a especificação da tabela 1 é utilizada como limite do valor a ser inserido.

Estado das entradas e dos relés:

No decorrer desse documento o estado das entradas e dos relés é representado por um valor numérico. Essa representação é mostrada na tabela 2.

Estado	Representação numérica
Entrada desatuada	**0**
Entrada atuada	**1**
Relé desligado	**0**
Relé ligado	**1**

Tabela 2 – Representação numérica do estado das entradas e dos relés.

Quando o comando para pulsar relé é enviada para a MIO, enquanto o relé estiver pulsando, a resposta para esse comando de ler o estado do relé será o mesmo valor do estado que foi passado no comando para pulsar o relé. Por exemplo, se um comando para pulsar 10 vezes o relé 1 é enviada para a MIO, com o estado do relé igual a ligado, então enquanto a MIO estiver pulsando o relé 1 essas 10 vezes o resultado para um comando de ler o estado do relé 1 será igual a relé ligado; após os 10 pulsos do relé 1 o resultado para um comando de ler o estado do relé 1 será igual a relé desligado.

Endereço IP: Por padrão de fábrica a MIO é configurada com o endereço IP 192.168.0.100.

Porta HTTP: Por padrão de fábrica a MIO é configurada com a porta 80 para as comunicações ethernet via protocolo HTTP.

A porta padrão do protocolo HTTP é a porta 80. Nos navegadores de internet, quando se faz uma comunicação via protocolo HTTP pela sua porta padrão (porta 80) a mesma pode ser omitida na URL.

Base de tempo da MIO: Os tempos relacionados às requisições de atuar sobre os relés (set_output) e configuração das entradas (set_input_config e get_input_config) são inseridos na base de tempo de 1ms, ou seja, a inserção de um valor de 100 em uma dessas requisições se refere ao tempo

de 100ms. Contudo, a MIO conta o tempo na base de 10ms, sendo assim, alguns valores podem sofrer arredondamentos. Os valores que não sejam múltiplos de 10ms são alterados da seguinte forma:

- Valores de 1ms até 9ms são alterados para 10ms;
- Valores maiores que 10ms, que não são múltiplos de 10ms, são alterados para o valor menor mais próximo e que é múltiplo de 10ms, ou seja, ele é arredondado para baixo;

A tabela abaixo mostra alguns exemplos de valores inserido no comando e seus valores adotados pela MIO.

Valor inserido	Valor adotado
1ms	10ms
5ms	10ms
59ms	50ms
127ms	120ms
2991ms	2990ms

Tabela 3 - Base de Tempo – Valor inserido x Valor adotado

Formatação de texto dos exemplos de resposta:

Neste documento, os exemplos de respostas que a MIO retorna para as requisições enviadas são exibidas em uma formatação de texto para melhor visualização e entendimento. No contexto prático, a MIO envia as respostas sem nenhuma formatação de texto, como quebras de linha ou espaços em branco entre os caracteres da resposta.

Todas as respostas que a MIO envia estão no formato JSON.

Relógio interno da MIO:

O relógio interno da MIO funciona no formato 24 horas (de 0 a 23 horas).

2.4 Requisições

2.4.1 Formatação de URLs:

Nos exemplos abaixo são demonstradas a construção de URLs, onde cada campo está demarcado por colchetes:

1. Exemplo de URL de requisição sem porta especificada, ou seja, usa a porta 80 (padrão do HTTP):

http://[endereço IP]/[comando]?[parâmetro 1]=[valor do parâmetro 1]&[parâmetro n]=[valor do parâmetro n].

2. Exemplo de URL de requisição com porta especificada:

http://[endereço IP]:[porta]/[comando]?[parâmetro 1]=[valor do parâmetro 1]&[parâmetro n]=[valor do parâmetro n].

Definição dos campos da URL de requisição:

Endereço IP: é o endereço de IP da MIO que se deseja estabelecer a comunicação ethernet via protocolo HTTP.

Porta: é a porta que está configurada na MIO para a comunicação ethernet via protocolo HTTP.

- **Comando:** é a função que se deseja executar na MIO. Os comandos disponíveis estão listados na tabela 4.

#	Comando	Descrição
1	set_output	Atuar sobre os relés.
2	get_input_status	Ler estado e transição das entradas.
3	get_output_status	Ler estado e transição dos relés.
4	get_io_status	Ler estado e transição das entradas e dos relés.
5	set_input_config	Configurar as entradas
6	get_input_config	Ler a configuração das entradas.
7	set_ip_config	Configurar os parâmetros de rede da MIO.
8	get_ip_config	Ler os parâmetros de rede da MIO.
9	get_device_info	Ler informações sobre a MIO.
10	get_rtc	Ler o relógio interno da MIO.
11	set_rtc	Configurar o relógio interno da MIO.

Tabela 4 – Comandos.

- **parâmetro**: alguns comandos necessitam de parâmetros para especificar, por exemplo, qual relé deve ser acionado na função set_output ou ainda se o relé deve ser ligado ou desligado. Os parâmetros podem ser obrigatórios ou opcionais.
- **valor do parâmetro**: quando for utilizado um parâmetro, então é obrigatório informar o seu valor. Cada parâmetro possui uma faixa de valor aceitável.

Resposta da MIO:

A MIO sempre retorna uma resposta com o resultado da execução do comando, sendo que a resposta está no formato JSON.

A resposta enviada pela MIO é formada pelos campos result e data.

O campo result pode ter as seguintes informações: success ou error. E o campo data pode conter as informações que foram solicitadas pelo comando ou não conter nenhuma informação.

Existem três possibilidades de resposta:

1. Resposta para confirmação de comando executado com sucesso, sem informações adicionais:

```
{
  “result”:“sucess”,
  “data”:null
}
```

2. Resposta para confirmação de comando executado com sucesso, com informações adicionais:

```
{
  “result”:“sucess”,
  “data”:[informações adicionais]
}
```

3. Resposta informando que houve um erro na execução de um comando:

```
{
  “result”:“error”,
  “data”:{
        “code”:[código de erro],
        “code_message”:“[mensagem de erro]”,
        “code_data”:[código do dado errado],
        “code_data_message”:“[mensagem do dado errado]”
   }
```

2.4.2 Definição de comandos:

Para os exemplos posteriores o endereço de IP da MIO utilizado será o 192.168.0.100.

2.4.2.1- Atuar sobre os relés:

- **Função**: set_output.
- **Descrição**: Comando usado para atuar sobre um dos relés da MIO.
- **Parâmetros**:

#	Nome	Descrição	Valores aceitos	Obrigatório
1	address	Relé a ser atuado.	1 ao nº máximo de relés da MIO (ver tabela 1)	Sim
2	state	Estado do relé.	0 = relé desligado 1 = relé ligado 2 = inverter estado do relé	Sim
3	time_1	Tempo que o relé permanece no estado definido pelo parâmetro state.	0 a 9999999 (ms)	Não
4	time_2	Tempo que o relé permanece no estado inverso ao definido pelo parâmetro state.	0 a 9999999 (ms)	Não
5	n_cycles	Quantidade de vezes que o relé pulsará.	1 a 999	Não

6	time_interval	Tempo que o relé permanece no estado inverso ao definido pelo parâmetro state, antes de repetir o ciclo de pulsos.	0 a 9999999 (ms)	Não
7	time_total	Tempo total que o relé ficará pulsando.	0 a 9999999 (ms)	Não

Tabela 5 – Parâmetros do comando set_output.

Através dos parâmetros relacionados na tabela 5 é possível gerar ações diferentes para os relés. As possíveis ações que podem ser geradas e a relação de parâmetros utilizados são mostradas na tabela 6. Qualquer comando de atuar sobre um relé que possuía uma combinação de parâmetros diferente das previstas na tabela 6 é considerado como um comando inválido e não é executado pelo MIO.

#	**Ação do relé**	**Parâmetros utilizados**	
1	Definir um estado para o relé assumir: ligado, desligado ou inverter o estado atual.	address	state
2	Gerar um pulso.	address state	time_1
3	Repetir um pulso por tempo indeterminado.	address state	time_1 time_2
4	Repetir um pulso pela quantidade de vezes definida.	address state time_1	time_2 n_cycles
5	Repetir um pulso por um tempo definido.	address state time_1	time_2 time_total
6	Repetir um ciclo de pulsos, com intervalo, por tempo indeterminado.	address state time_1	time_2 n_cycles time_interval
7	Repetir um ciclo de pulsos, com intervalo, por um tempo definido.	address state time_1 time_2	n_cycles time_interval time_total

Tabela 6 – Ação do relé x Parâmetros utilizados.

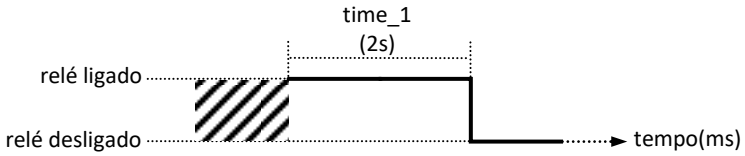
Nos comandos que o parâmetro time_total é utilizado, o valor dele deve ser igual ou maior que a soma dos valores dos parâmetros de tempo usados nesse mesmo comando.

Exemplos do comando:

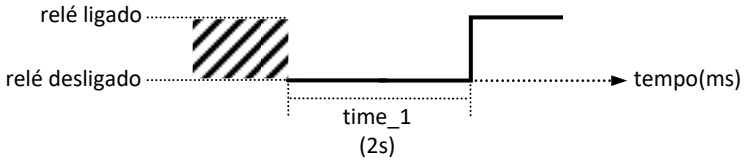
- Ligar o relé 1:
  http://192.168.0.100/set_output?address=1&state=1.

- Desligar o relé 1:
  http://192.168.0.100/set_output?address=1&state=0.

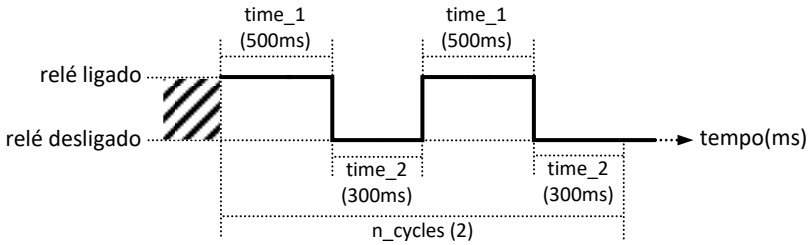
- Ligar o relé 3 por 2s:
  http://192.168.0.100/set_output?address=3&state=1&time_1=2000.



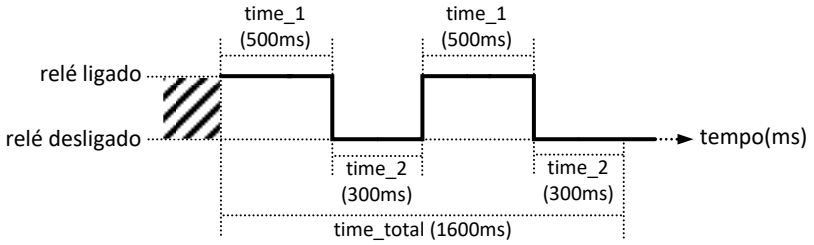
- Desligar o relé 3 por 2s:
  http://192.168.0.100/set_output?address=3&state=0&time_1=2000.



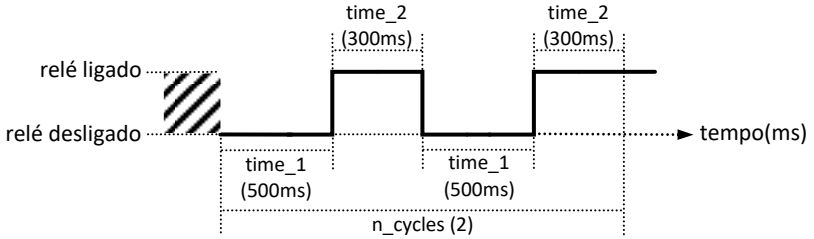
- Pulsar o relé 4, permanecendo ligado por 500ms e desligado por 300ms. Repete esse pulso 2 vezes (utiliza o parâmetro n_cycles):
  http://192.168.0.100/set_output?address=4&state=1&time_1=500&time_2=300&n_cycles=2.



- Pulsar o relé 4, permanecendo ligado por 500ms e desligado por 300ms. Repete esse pulso 2 vezes (utiliza o parâmetro time_total):
  http://192.168.0.100/set_output?address=4&state=1&time_1=500&time_2=300&time_total=1600.

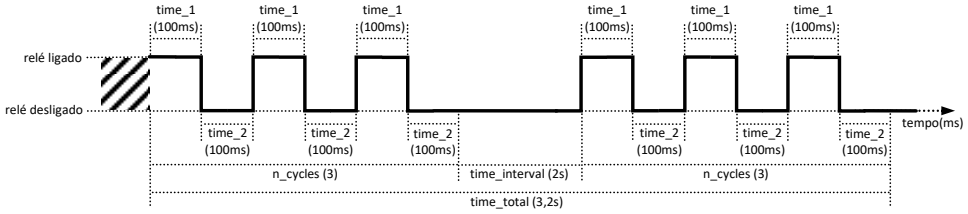


- Pulsar o relé 4, permanecendo desligado por 500ms e ligado por 300ms. Repete esse pulso 2 vezes (utiliza o parâmetro n_cycles):
  http://192.168.0.100/set_output?address=4&state=0&time_1=500&time_2=300&n_cycles=2.

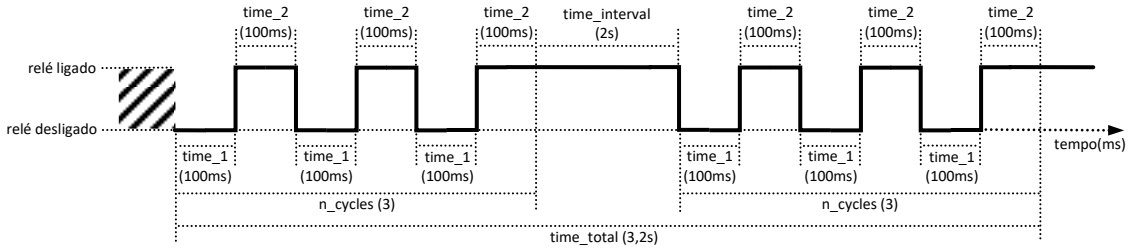


- Pulsar o relé 1, permanecendo ligado por 2s e desligado por 1s. Repete esse ciclo por um tempo indeterminado:
  http://192.168.0.100/set_output?address=1&state=1&time_1=2000&time_2=1000.

- Pulsar 3 vezes o relé 1, permanecendo ligado por 100ms, desligado por 100ms e com um intervalo de 2s entre os ciclos de pulsos. Repete esse ciclo por um tempo de 3,2s:
  http://192.168.0.100/set_output?address=1&state=1&time_1=100&time_2=100&n_cycles=3&time_interval=2000&time_total=3200.



- Pulsar 3 vezes o relé 1, permanecendo desligado por 100ms, ligado por 100ms e com um intervalo de 2s entre os ciclos de pulsos. Repete esse ciclo por um tempo de 3,2s:
  http://192.168.0.100/set_output?address=1&state=0&time_1=100&time_2=100&n_cycles=3&time_interval=2000&time_total=3200.



- Pulsar 3 vezes o relé 1, permanecendo ligado por 100ms, desligado por 100ms e com um intervalo de 2s entre os ciclos de pulsos. Repete esse ciclo por um tempo indeterminado:

  http://192.168.0.100/set_output?address=1&state=1&time_1=100&time_2=100&n_cycles=3&time_interval=2000

- **Resposta do comando**:
  A MIO envia uma resposta de confirmação que o comando foi executado, sem informações adicionais.

- **Exemplo de código PHP para o comando**:
  Pulsar o relé 4 duas vezes, permanecendo ligado por 500ms e desligado por 300ms.

```
$param = "address=1&state=1&time_1=500&time_2=300&time_total=1600";
$url = "http://192.168.0.100/set_output?".$param;
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.2- Ler estado e transição das entradas:

- **Comando**: get_input_status.
- **Descrição**: Comando usado para ler o estado de todas as entradas da MIO e se ocorreu alguma transição no estado das entradas desde a última leitura.
- **Parâmetros**:

#	Nome	Descrição	Valores aceitos	Obrigatório
1	format	Formato da resposta da MIO para esse comando. Se o parâmetro não for	0 = formato campos 1 = formato decimal	Não

		utilizado, a MIO responderá no formato campos.	2 = formato binário	

Tabela 7 – Parâmetros do comando get_input_status.

- **Formato campos**:
  32 números decimais, separados por vírgulas, são usados para representar as entradas. Cada número decimal representa uma entrada, sendo que o número mais a direita representa a entrada 1 e assim sucessivamente.

- **Formato decimal**:
  Apenas 1 número decimal representa todas as entradas. Cada entrada tem um peso na composição desse valor, esse peso é definido por: $(2^{(i-1)})*(k)$; onde i é o número da entrada, por exemplo, para a entrada 3 o valor de i é 3; e k é o estado da entrada, sendo k=0 para a entrada desatuada ou sem transição e k=1 para entrada atuada ou com transição. O número decimal que representa todas as entradas é o somatório do peso de todas as entradas da MIO.

- **Formato binário**:
  32 caracteres são usados para representar todas as entradas. Cada caractere representa uma entrada, sendo que o caractere mais a direita representa a entrada 1 e assim sucessivamente.

- **Exemplos do comando**:
  Para os exemplos abaixo é considerado que as entradas 1 e 4 estão atuadas e o restante das entradas estão desatuadas:

  1 - Ler o estado das entradas sem enviar o parâmetro *format*, ou seja, no formato campos:
  http://192.168.0.100/get_input_status.

  2 - Ler o estado das entradas no formato decimal:
  http://192.168.0.100/get_input_status?format=1.

  3 - Ler o estado das entradas no formato binário:
  http://192.168.0.100/get_input_status?format=2.

- **Resposta do comando (de acordo com os exemplos acima)**:
  A resposta para esse comando é composta por 2 campos: *state* e *transition*. O campo *state* mostra o estado atual das entradas, sendo que 0 representa entrada atuada e 1 entrada desatuada. O campo *transition* mostra se houve uma mudança no estado da entrada, uma transição, desde a última leitura da mesma, sendo que 0 representa que não ocorreu uma mudança na entrada e 1 que ocorreu uma mudança na entrada.
  Ambos os campos mostram os resultados de acordo com o valor do parâmetro *format*.

1- Ler o estado das entradas sem enviar o parâmetro *format*, ou seja, no formato campos:

```
{
  “result”:“sucess”,
```

```
    “data”:{
        “inputs”:{
            “state”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1],
            “transition”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1]
        }
    }
}
```


2- Ler o estado das entradas no formato decimal:


```
{
    “result”:“sucess”,
    “data”:{
        “inputs”:{
            “state”:9,
            “transition”:9
        }
    }
}
```


3- Ler o estado das entradas no formato binário:


```
{
    “result”:“sucess”,
    “data”:{
        “inputs”:{
            “state”:“00000000000000000000000000001001”,
            “transition”:“00000000000000000000000000001001”
        }
    }
}
```

- **Exemplo de código PHP para o comando**:

```
$url = “http://192.168.0.100/get_input_status”;
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.3- Ler estado e transição dos relés:

- **Comando**: get_output_status.
- **Descrição**: Comando usado para ler o estado de todos os relés da MIO e se ocorreu alguma transição no estado dos relés desde a última leitura.
- **Parâmetros**: mesmos parâmetros usado no comando para ler todas as entradas da MIO (get_input_status).
- **Exemplos do comando**:
  Para os exemplos abaixo é considerado que os relés 2, 3 e 8 estão ligados e o restante dos relés estão desligados:

  1 - Ler o estado dos relés sem enviar o parâmetro format, ou seja, no formato campos:
  http://192.168.0.100/get_output_status.

  2 - Ler o estado dos relés no formato decimal:
  http://192.168.0.100/get_output_status?format=1.

  3 - Ler o estado dos relés no formato binário:
  http://192.168.0.100/get_output_status?format=2.

- **Resposta do comando (de acordo com os exemplos acima)**:
  A resposta para esse comando segue o mesmo formato da resposta ao comando get_input_status, sendo que difere apenas no nome de um dos campos que foi alterado de inputs para outputs.

1- Ler o estado dos relés sem enviar o parâmetro format, ou seja, no formato campos:


```
{
  “result”:“sucess”,
  “data”:{
    “outputs”:{
      “state”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0],
      “transition”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0]
    }
  }
}
```


2- Ler o estado dos relés no formato decimal:


```
{
  “result”:“sucess”,
  “data”:{
    “outputs”:{
      “state”:134,
      “transition”:134
    }
  }
}
```


3- Ler o estado dos relés no formato binário:


```
{
  “result”:“sucess”,
  “data”:{
    “outputs”:{
      “state”:“00000000000000000000000010000110”,
      “transition”:“00000000000000000000000010000110”
    }
  }
}
```


- **Exemplo de código PHP para o comando**:

```
$url = “http://192.168.0.100/ get_output_status”;
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.4- Ler estado e transição das entradas e dos relés:

- **Comando**: get_io_status.
- **Descrição**: Comando usado para ler o estado de todas as entradas e de todos as saídas da MIO e também verificar se ocorreu alguma transição no estado das entradas ou saídas desde a última leitura.
- **Parâmetros**: mesmos parâmetros usados no comando para ler todas as entradas da MIO (get_input_status).
- **Exemplos do comando**:
  Para os exemplos abaixo é considerado que as entradas 1 e 4 estão atuadas e o restante das entradas estão desatuadas; e que os relés 2, 3 e 8 estão ligados e o restante dos relés estão desligados:

  1 - Ler o estado das entradas e dos relés sem enviar o parâmetro format, ou seja, no formato campos:
  http://192.168.0.100/get_io_status.

  2 - Ler o estado das entradas e dos relés no formato decimal:
  http://192.168.0.100/get_io_status?format=1.

  3 - Ler o estado das entradas e dos relés no formato binário:
  http://192.168.0.100/get_io_status?format=2.

- **Resposta do comando (de acordo com os exemplos acima)**:
  A resposta para esse comando segue o mesmo formato da resposta ao comando get_input_status, sendo que difere apenas na inclusão de um novo campo chamado de outputs.

1- Ler o estado das entradas e dos relés sem enviar o parâmetro format, ou seja, no formato campos:


```
{
   “result”:“sucess”,
   “data”:{
      “inputs”:{
         “state”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1],
         “transition”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1]
      },
      “outputs”:{
         “state”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0],
         “transition”:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0]
      }
   }
}
```


2- Ler o estado das entradas e dos relés no formato decimal:


```
{
   “result”:“sucess”,
   “data”:{
      “inputs”:{
```

```
            "state":9,
            "transition":9
        },
        "outputs":{
            "state":134,
            "transition":134
        }
    }
}
```


3- Ler o estado das entradas e dos relés no formato binário:


```
{
    "result":"sucess",
    "data":{
        "inputs":{
            "state":"00000000000000000000000000001001",
            "transition":"00000000000000000000000000001001"
        },
        "outputs":{
            "state":"00000000000000000000000010000110",
            "transition":"00000000000000000000000010000110"
        }
    }
}
```


- **Exemplo de código PHP para o comando**:

```
$url = "http://192.168.0.100/get_io_status";
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.5- Configurar as entradas:

- **Comando**: set_input_config.
- **Descrição**: Comando usado para configurar as entradas da MIO.
- **Parâmetros**:

#	Nome	Descrição	Valores aceitos	Obrigatório
1	address	Entrada a ser configurada.	1 ao nº máximo de entradas da MIO (ver tabela 1)	Sim
2	enable	Habilita ou desabilita a entrada.	0 = entrada desabilitada 1 = entrada habilitada	Sim
3	delay_on	Tempo que a entrada demora em assumir que seu estado mudou de desatuada para atuada.	0 a 999999 (ms)	Não
4	delay_off	Tempo que a entrada demora em assumir que seu estado mudou de atuada para desatuada.	0 a 999999 (ms)	Não

Tabela 8 – Parâmetros do comando set_input_config.

- **Observações**:

Parâmetro	Observações
enable	Uma entrada desabilitada sempre informa seu estado como desatuada. Por padrão de fábrica todas as entradas estão configuradas como habilitadas.
delay_on	Por padrão de fábrica esse tempo está configurado com 100ms.
delay_off	Por padrão de fábrica esse tempo está configurado com 100ms.

Tabela 9 – Observações do comando set_input_config.

- **Exemplos do comando:**
  - Desabilitar a entrada 1: http://192.168.0.100/set_input_config?address=1&enable=0.
  - Habilitar a entrada 1: http://192.168.0.100/set_input_config?address=1&enable=1.
  - Alterar o delay_on para 1s e o delay_off para 1s: http://192.168.0.100/set_input_config?address=1&enable=1&delay_on=1000&delay_off=1000.

- **Resposta do comando:**

  A MIO envia uma resposta de confirmação que o comando foi executado, sem informações adicionais.

- **Exemplo de código PHP para o comando:**

  Para a entrada 1, alterar o delay_on para 5s e o delay_off para 5s.

```
$param = "address=1&enable=1&delay_on=5000&delay_off=5000";
$url = "http://192.168.0.100/set_input_config?".$param;
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.6- Ler a configuração das entradas:

- **Comando**: get_input_config.
- **Descrição**: Comando usado para ler a configuração de uma das entradas da MIO.
- **Parâmetros**:

#	Nome	Descrição	Valores aceitos	Obrigatório
1	address	Entrada que será lida a configuração.	1 ao nº máximo de entradas da MIO (ver tabela 1)	Sim

Tabela 10 – Parâmetros do comando get_input_config.

- **Exemplos do comando:**
  - Ler a configuração da entrada 1: http://192.168.0.100/get_input_config?address=1.

- **Resposta do comando (de acordo com o exemplo acima):**
  Considerando que a entrada 1 esteja habilitada, com o delay_on=100ms e com o delay_off =100ms.

```
{
  “result”:“sucess”,
  “data”:{
      “address”:1,
      “enable”:1,
      “delay_on”:100,
      “delay_off”:100
  }
}
```

- **Exemplo de código PHP para o comando:**
  Ler as configurações da entrada 1.

```
$param = "address=1";
$url = "http://192.168.0.100/get_input_config?".$param;
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.7- Configurar os parâmetros de rede da MIO:

- **Comando**: set_ip_config.
- **Descrição**: Comando usado para configurar alguns parâmetros da rede ethernet da MIO. Após a execução desse comando a MIO será reinicializada para assumir os novos parâmetros configurados.
- **Parâmetros**:

#	Nome	Descrição	Valores aceitos	Obrigatório
1	ip	Endereço IP da MIO.	No formato: nnn.nnn.nnn.nnn onde nnn varia de 0 a 255.	Sim
2	port	Porta que a MIO utiliza para se comunicar via protocolo HTTP.	1 até 65535	Sim
3	mask	Sub máscara de rede da MIO.	No formato: nnn.nnn.nnn.nnn onde nnn varia de 0 a 255.	Sim
4	gateway	Endereço de IP do gateway.	No formato: nnn.nnn.nnn.nnn onde nnn varia de 0 a 255.	Sim

Tabela 11 – Parâmetros do comando set_ip_config.

- **Exemplos do comando:**
  - Alterar o endereço de IP da MIO para 192.168.0.101, a porta para 5091, a sub máscara de rede para 255.255.255.0 e o endereço de IP do gateway para 192.168.0.2: http://192.168.0.100/set_ip_config?ip=192.168.0.101&port=5091&mask=255.255.255.0&gateway=192.168.0.2.

- **Resposta do comando:**

  A MIO envia uma resposta de confirmação que o comando foi executado, sem informações adicionais.

- **Exemplo de código PHP para o comando:**

  Alterar o endereço de IP da MIO para 192.168.0.100, a porta para 80, a sub máscara de rede para 255.255.255.0 e o endereço de IP do gateway para 192.168.0.1.

```
$param =
"ip=192.168.0.100&port=80&mask=255.255.255.0&gateway=192.168.0.1";
$url = "http://192.168.0.100/set_ip_config?".$param;
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.8- Ler os parâmetros de rede da MIO:

- **Comando**: get_ip_config.
- **Descrição**: Comando usado para ler os parâmetros da rede ethernet que a MIO está utilizando.
- **Parâmetros**: Este comando não possui parâmetros.
- **Exemplo do comando**: http://192.168.0.100/get_ip_config.
- **Resposta do comando:**

```
{
   “result”:“sucess”,
   “data”:{
         “source_ip”:“192.168.0.100”,
“source_port”:80,
“subnet_mask”:“255.255.255.0”,
“gateway_ip”:“192.168.0.1”,
 “mac_address”:“00-04-A3-3D-5F-91”
    }
}
```

- **Exemplo de código PHP para o comando:**

```
$url = "http://192.168.0.100/get_ip_config”;
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
```

```
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.9- Ler informações sobre a MIO:

- **Comando**: get_device_info.
- **Descrição**: Comando usada para ler informações sobre a MIO, sendo elas: modelo da MIO, código do modelo, versão do firmware e ID único da MIO. A relação entre o modelo da MIO e o seu código está representado na tabela 12.

Modelo da MIO	Código do modelo
MIO402	40
MIO400	31
MIO800	32
MIO2408	34
MIO0816	41

Tabela 12 – Modelo MIO x Código do modelo.

- **Parâmetros**: Este comando não possui parâmetros.
- **Exemplo do comando**: http://192.168.0.100/get_device_info.
- **Resposta do comando:**

```
{
   “result”:“sucess”,
   “data”:{
         “model”:“MIO800”,
         “model_code”:32,
“version_firmware”:“3.02”,
“unique_id”:2738708369
    }
}
```

- **Exemplo de código PHP para o comando:**

```
$url = "http://192.168.0.100/get_device_info";
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
```

```
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.10- Ler o relógio interno:

- **Comando**: get_rtc.
- **Descrição**: Comando usado para ler as informações de data e hora do relógio interno da MIO.
- **Parâmetros**: Este comando não possui parâmetros.
- **Exemplo do comando**: http://192.168.0.100/get_rtc.
- **Resposta do comando:**

- A data é retornada no formato yy-mm-dd, onde yy é o ano, mm é o mês e dd é o dia.

- A hora é retornada no formato hh:mm:ss, onde hh é a hora, mm são os minutos e ss são os segundos.

```
{
   “result”:“sucess”.
   “data”:{
         “date”:”18-4-16”,
         “time”:“11:57:35”
    }
}
```

- **Exemplo de código PHP para o comando:**

```
$url = "http://192.168.0.100/get_rtc";
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

2.4.2.11- Configurar o relógio interno:

- **Comando**: set_rtc.
- **Descrição**: Comando usado para configurar o relógio interno da MIO.
- **Parâmetros**:

#	Nome	Descrição	Valores aceitos	Obrigatório
1	date	Define a data (ano, mês, dia) para o relógio interno da MIO.	No formato: yy-mm-dd onde yy é o ano, mm é o mês e dd é o dia.	Sim
2	time	Define o horário (hora, minutos, segundos) para o relógio interno da MIO.	No formato: hh:mm:ss onde hh é a hora, mm são os minutos e ss são os segundos.	Sim

Tabela 13 – Parâmetros do comando set_rtc.

- **Exemplos do comando:**
  - Alterar a data da MIO para 30 de janeiro de 2018 e o horário para 12 horas 30 minutos e 55 segundos: http://192.168.0.100/set_rtc?date=18-01-30&time=12:30:55.
- **Resposta do comando:**
  A MIO envia uma resposta de confirmação que o comando foi executado, sem informações adicionais.
- **Exemplo de código PHP para o comando:**
  Alterar a data da MIO para 24 de abril de 2018 e o horário para 19 horas 5 minutos e 35 segundos.

```
$param = "date=18-04-24&time=19:5:35";
$url = "http://192.168.0.100/set_rtc?".$param;
$ch = curl_init ();
curl_setopt ($ch, CURLOPT_URL, $url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt ($ch, CURLOPT_HTTPHEADER, array ('Accept: application/json'));
$response = curl_exec ($ch);
curl_close ($ch);
echo $response;
```

Resumo dos comandos:

A tabela 14 mostra um resumo dos comandos, parâmetros e valores aceitáveis dos parâmetros.

#	Descrição	Comando	Parâmetros	
			Nome	Valores aceitáveis
1	Atuar sobre os relés	set_output	address	1 ao nº máximo de relés da MIO (ver tabela 1)
			state	0=desligado; 1=ligado; 2=inverter
			time_1	0 a 9999999 (ms)
			time_2	0 a 9999999 (ms)
			n_cycles	1 a 999
			time_interval	0 a 9999999 (ms)
			time_total	0 a 9999999 (ms)
2	Ler estado e transição das entradas	get_input_status	format	0=campos; 1=decimal; 2=binário
3	Ler estado e transição dos relés	get_output_status	format	0=campos; 1=decimal; 2=binário
4	Ler estado e transição das entradas e dos relés	get_ios_status	format	0=campos; 1=decimal; 2=binário
5	Configurar as entradas	set_input_config	address	1 ao nº máximo de entradas da MIO (ver tabela 1)
			enable	0=desabilita entrada 1=habilita entrada
			delay_on	0 a 9999999 (ms)
			delay_off	0 a 9999999 (ms)
6	Ler configuração das entradas	get_input_config	address	1 ao nº máximo de entradas da MIO (ver tabela 1)
7	Configurar os parâmetros de rede da MIO	set_ip_config	ip	nnn.nnn.nnn.nnn (nnn = 0 a 255)
			port	1 a 65535
			mask	nnn.nnn.nnn.nnn (nnn = 0 a 255)
			gateway	nnn.nnn.nnn.nnn (nnn = 0 a 255)
8	Ler os parâmetros de rede da MIO	get_ip_config		
9	Ler informações sobre a MIO	get_device_info		
10	Ler o relógio interno da MIO	get_rtc		
11	Configurar o relógio interno da MIO	set_rtc	date	yy-mm-dd (yy=ano; mm=mês; dd=dia)
			time	hh:mm:ss (hh=hora; mm=minuto; ss=segundos)

Tabela 14 – Comandos, parâmetros e valores aceitáveis.

Resposta de erro

Na resposta de erro, para o caso de algum erro na linha de comando, os possíveis códigos de erro e suas mensagens são mostrados na tabela 15. Os campos dessa tabela preenchidos com colchetes variam, pois mostram a mensagem recebida pela MIO e que são desconhecidos para a mesma.

code	code_message	code_data	code_data_ message
1	Method not allowed	null	null
2	Unknown command	1	*[comando recebido]*
3	Unknown parameter	1	*[parâmetro recebido]*
4 5 6	Missing parameter Duplicate parameter Invalid parameter value	10	address
		11	state
		12	time_1
		13	time_2
		14	n_cycles
		15	time_interval
		16	time_total
		17	format
		18	enable
		19	delay_on
		20	delay_off
		30	ip
		31	port
		32	mask
		33	gateway
		40	rtc date
		41	rtc time
100	Unknown error	null	null

Tabela 15 – Resposta de Erro.

3. Integração via protocolo BACnet

3.1 Introdução BACnet

A MIO v3, a partir da versão de firmware 3.10, está apta a comunicar-se através do protocolo BACnet (Building Automation Control Network). É um protocolo padrão utilizado em automações prediais, comumente chamados de sistemas BMS (Building Management System).

Neste capítulo estão descritos os recursos BACnet disponíveis no produto. Recomendamos a leitura dessas informações antes de iniciar a utilização do equipamento.

3.2 Especificações BACnet

Na MIO v3 está disponível o BACnet/IP, que utiliza a rede IP para comunicação, e está em acordo com a norma ANSI/ASHARE 135-2016.

De acordo com a norma, os recursos disponíveis na MIO v3 a categorizam como um controlador perfil B-ASC.

3.3 Recursos

3.3.1 Serviços

Serviços são as mensagens trocadas entre dispositivos e/ou softwares com objetivo de enviar comandos e respostas. Na MIO v3 estão implementados os seguintes serviços:

- **Who is / I am** – comandos básicos de identificação do dispositivo na rede BACnet. Ele é utilizado na descoberta de equipamentos pelo software gerente e quando o equipamento é conectado na rede BACnet.
- **Read Property –** comando utilizado para leitura de propriedades de objetos. Ele é utilizado, por exemplo, para ler os estados das entradas da MIO.
- **Write Property** – comando utilizado para escrita em propriedades de objetos. Na MIO, ele é utilizado, por exemplo, para modificar o estado de suas saídas.
- **Subscribe COV –** comando utilizado para configurar o envio de notificações quando ocorrer uma alteração na propriedade de um objeto. Na MIO v3, a notificação é gerada a partir de uma alteração na propriedade Present Value de um dos objetos abaixo:
  - **Binary Inputs –** quando atuar e desatuar um input;
  - **Binary Output** – quando um output for atuado ou desatuado a partir da Lógica Programável (**LP**). Mudança nos outpus por outros comandos não gera notificação;
  - **Counter** – quando a LP alterar o valor do contador;
  - **Flag -** quando a LP alterar o valor da flag;
  - **Timer –** quando o timer chegar ao tempo carregado na LP.

O equipamento suporta até 50 inscrições simultâneas, e ao receber uma inscrição com o campo “life time” igual a zero, a configuração é salva em memória. Desta forma, mesmo que o equipamento seja desligado, a configuração será mantida.

- **Confirmed COV Notification –** é a mensagem enviada pelo equipamento para notificar a alteração da propriedade do objeto. Esta mensagem requer uma confirmação de recebimento do assinante.
- **Unconfirmed COV Notification –** é a mensagem enviada pelo equipamento para notificar a alteração da propriedade do objeto. Esta mensagem não requer uma confirmação de recebimento do assinante.

3.3.2 Objetos

Objetos são funcionalidades que podem ser acessadas no equipamento. Cada objeto possui um conjunto de propriedades e todas podem ser lidas. No entanto somente algumas delas suportam edição, que podem ser realizadas através do software gerente ou através do Configurador MIO.

Na MIO v3 estão disponíveis os seguintes objetos:

- **Device** – é o objeto que contém informações que categorizam o dispositivo. As principais propriedades deste objeto são:
  - **Object Identifier** – é uma identificação numérica do dispositivo. Ela deve ser única na rede BACnet.
  - **Object Name** – é o nome do dispositivo, que também deve ser único da rede BACnet.
  - **Vendor Name** – é a identificação do fabricante.
  - **Vendor ID** – é a identificação numérica do fabricante (1239).
  - **Model Name –** é o modelo do dispositivo.
  - **Max APDU Lenght Accepted** – Na MIO, a quantidade máxima de dados de aplicação é 1476 bytes.
  - **Segmentation Supported** – A MIO não suporta segmentação.

- **Binary Input** – é o objeto que define os inputs da MIO. Para cada input da MIO, há um objeto Binary Input com suas respectivas propriedades. Entre elas, as principais são:
  - **Object Identifier** – é um número sequencial que identifica o input da MIO. Por exemplo, 1 corresponde ao Input 1, 2 ao Input 2...
  - **Object Name** – é o nome do input. Ele deve ser único na MIO. Por default é Input 1, Input 2....
  - **Present Value** – é o estado do input e pode ser 1 ou 0.
  - **Polarity** – indica a lógica de interpretação do objeto. Na MIO essa propriedade é somente leitura e o padrão é 0. Isso indica que quando o Present Value é 1, o input está atuado.

- **Binary Output** – é o objeto que define os outputs da MIO. Para cada output da MIO, há um objeto Binary Output com suas respectivas propriedades. Entre elas, as principais são:
  - **Object Identifier** – é um número sequencial que identifica o output da MIO. Por exemplo, 1 corresponde ao Output 1, 2 ao Output 2...

- **Object Name** – é nome do output. Ele deve ser único na MIO. Por default é Output 1, Output 2....
- **Present Value** – é o estado do output e pode ser 1 ou 0.
- **Polarity** – indica a lógica de interpretação do objeto. Na MIO essa propriedade é somente leitura e o padrão é 0. Isso indica que quando o Present Value é 1, o output está atuado.

- **Counter –** é um objeto proprietário relacionado ao contador da LP. O objeto permite a leitura do contador e suas propriedades são:
  - **Object Identifier** – é um número sequencial que identifica o contador da MIO. Por exemplo, 1 corresponde ao Counter 1, 2 ao Counter 2...
  - **Object Name** – é nome do contador. Ele deve ser único na MIO. Por default é Counter 1, Counter 2....
  - **Object Type –** valor que identifica o tipo de objeto. Para o counter é 130.
  - **Present Value –** é o valor disponível no contador, que vai de 0 a 1 milhão. Esse valor é um dado tipo Unsigned Integer (32 bits).

- **Flag** – é um objeto proprietário relacionado à flag da LP. O objeto permite a leitura da flag e suas propriedades são:
  - **Object Identifier** – é um número sequencial que identifica a flag da MIO. Por exemplo, 1 corresponde a Flag 1, 2 a Flag 2...
  - **Object Name** – é nome da flag. Ele deve ser único na MIO. Por default é Flag 1, Flag 2....
  - **Object Type –** valor que identifica o tipo de objeto. Para a Flag é 131.
  - **Present Value –** é o valor que indica o estado da Flag, é um dado do tipo Enumerated.

- **Timer** – é um objeto proprietário relacionado ao timer da LP. O objeto permite a leitura do timer e suas propriedades são:
  - **Object Identifier** – é um número sequencial que identifica o timer da MIO. Por exemplo, 1 corresponde ao Timer 1, 2 ao Timer 2...
  - **Object Name** – é nome do timer. Ele deve ser único na MIO. Por default é Timer 1, Timer 2....
  - **Object Type –** valor que identifica o tipo de objeto. Para o Timer é 132.
  - **Present Value –** é o valor disponível no timer, que indica o tempo em segundos. É um dado do tipo Unsigned Integer (32 bits).

Tabela 1 – Resumo de objetos e propriedades

Objeto	Propriedade	Valor padrão	Permite edição	Dados suportados	COV
Device	Object Identifier	ID da MIO	Sim	1 até 4194302	
	Object Name	MIO<modelo>-<ID da MIO>	Sim	32 caracteres	
	Vendor Name	Commbox Tecnologia	Não		
	Vendor ID	1239	Não		
	Model Name	MIO400, MIO402, MIO800, MIO0816, MIO2408	Não		
	Max APDU Lenght Accepted	1476 bytes	Não		
	Segmentation Supported	Não	Não		
Binary Input	Object Identifier		Não		
	Object Name	Input i, onde i é o Object Identifier	Sim	28 caracteres	
	Present Value	0	Não		Sim
	Polarity	0	Não		
Binary Output	Object Identifier	Valor numérico	Não		
	Object Name	Output i, onde i é o Object Identifier	Sim	28 caracteres	
	Present Value	0	Sim	1 ou 0	Sim LP
	Polarity	0	Não		
Counter	Object Identifier	Valor numérico	Não		
	Object Name	Counter i, onde i é o Object Identifier	Sim	28 caracteres	
	Present Value	0	Não	Unsigned Integer (32 bits)	Sim LP
	Object Type	130	Não		
Flag	Object Identifier	Valor numérico	Não		
	Object Name	Flag i, onde i é o Object Identifier	Sim	28 caracteres	
	Present Value	0	Não	Enumerated	Sim LP
	Object Type	131	Não		
Timer	Object Identifier	Valor numérico	Não		
	Object Name	Timer i, onde i é o Object Identifier	Sim	28 caracteres	
	Present Value	0	Não	Unsigned Integer (32 bits)	Sim LP
	Object Type	132	Não		

Configuração

Para configuração de propriedades de rede, consulte o manual do Configurador MIO. É uma aplicação Commbox destinada a configuração do equipamento.

4. Protocolo OnVif

A Multi I/O v3 pode ser integrada através do protocolo OnVif. Com a integração é possível realizar a execução das funcionalidades abaixo:

- Descoberta na rede;
- Informações do equipamento (modelo, versão do FW, quantidade de inputs/outputs, etc.);
- Leitura de Inputs;
- Atuação de outputs.

5. Protocolo de aplicação COMMBOX

5.1 Definições e considerações gerais

- **Ordem de transmissão/composição do frame/variáveis:**

MSB transmitido em primeiro lugar (posição mais à esquerda do campo/variável)

ex.: 1 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1
MSB (bit 15) LSB (bit 0)

- Valores padrão:

- ON=1, OFF = 0
- Sim(S) = 1, Não(N) = 0
- Bin_data (binário): dados são acessados a nível de bitfield

ex.: 1 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1
bit .15 bit .10 bit .0

- Tamanho dos campos/variáveis:

- Byte/char = 1 byte (8 bits)
- Int = 2 bytes (16 bits)
- Long = 4 bytes (32 bits)
- String = string, incluindo o terminador 0x00 (valor zero em hexadecimal)
- Array(x) = vetor de bytes, tem o tamanho de "x" bytes

- Conotação de bases numéricas:

- 0x = base hexadecimal
- não especificado = base decimal

- Endereços:

- Valores entre 1 e *n*, salvo especificação em contrário.

5.2 Protocolo de Comunicação

5.2.1 Composição do frame:

O frame é dividido em duas partes: Header e dados da aplicação
Header do Frame / **Dados da Aplicação**

Campo	Tamanho (bytes)	Descrição	Exemplo
1	Long (4)	Header Symbol	0xAA55AA55
2	Long (4)	Protocol Version	0x00000001

3	Long (4)	Data Size	0x00000006
4	Long (4)	Source Id	0x0234098E
5	Long (4)	Destiny Id	0x04352AF5
6	Long (4)	Frame Sequence	0x00000005
7	Long (4)	Checksum	0xFF4901EB
8	Long (4)	Opcode	0x0000002A
9	data array (campo de dados)	Application Data	...

- **Header Symbol:** Constante identificadora de início de frame, igual a 0xAA55AA55.
- **Protocol Version:** Identifica a versão do protocolo.
- **Data Size:** Especifica o tamanho em bytes do campo "Application Data".
- **Source Id:** Identifica o Id do equipamento que está transmitindo o frame.
- **Destiny Id:** Identifica o Id do equipamento que irá receber o frame.
- **Frame Sequence:** Identifica o número seqüencial do frame.
- **Checksum:** Contempla a soma de TODOS os bytes do frame, com exceção do próprio campo checksum.
- **Opcode:** Identifica qual procedimento/comando será executado com os dados fornecidos.
- **Application Data:** Contém os dados relacionados ao Opcode.

5.2.2 ACK/NACK:

A grande maioria dos opcodes exige o retorno de ack/nack para encerramento do processo.

A codificação para ack/nack ocorre da seguinte forma:

- ACK: Opcode + 0x80000000 (opcode com bit 31 setado)
- NACK: Opcode + 0x40000000 (opcode com bit 30 setado)

O NACK informa, via campo de dados, as seguintes informações:
1- Código do Erro (Error Code): Long
2- Dados do Erro (Error Data): Long

5.3 Opcodes

As informações abaixo se referem aos dados a serem inseridos dentro do campo "Dados da Aplicação" (Application Data), de acordo com os respectivos opcodes.

5.3.1 Opcode 01- Atuação nas saídas:

Requisição:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	Long (4)	Endereço	1	Especifica o endereço da Saída
2	Byte (1)	ON/OFF	ON	Especifica o tipo de atuação
3	Long (4)	Tempo_total	3000	Especifica o tempo total da atuação na saída. Unidade de tempo = 10ms (0,01s)
4	Long (4)	Tempo_semiciclo1	100	Especifica o "T_ON" da atuação na saída. Unidade de tempo = 10ms (0,01s)
5	Long (4)	Tempo_semiciclo2	200	Especifica o "T_OFF" da atuação na saída. Unidade de tempo = 10ms (0,01s)
6	Byte (1)	Memoriza em flash	S/N	Memoriza a atuação em memória não volátil

Temporização:

Quando temporizado, os campos 4 e 5 podem ser usados para gerar pulsos. Isto ocorre quando a soma destes campos é menor que o tempo total.

Isto permite, por exemplo, que um relé ligue por 1 segundo e desligue por 2 segundos durante um determinado período de tempo, definido por "Tempo total" (conforme exemplo acima).

Ao expirar o tempo total, o relé permanece na condição informada pelo campo ON/OFF, independente do tempo ligado/desligado ou estado atual.

Memorização da atuação:

Geralmente utilizado para atuação não temporizadas (one shot), este recurso permite que após a queda de energia, o relé volte a condição estipulada no frame, uma vez que esta condição fora armazenada em memória não volátil (flash).

ATENÇÃO:

- Em função das características da memória flash, esta operação não pode ser executada mais de 100 mil vezes.
- Esta operação demanda maior tempo de execução, no caso 6ms (0,006s) ao invés de 300us (0,0003s).

Resposta:
ACK / NACK

5.3.2 Opcode 02- Leitura do estado das saídas:

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

Resposta:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	Long (4)	Output_data	0x0000000F saídas 1 a 4 ligadas, 5 a 32 desligadas	Reflete o estado das saídas. Bit .0 = saída 1, Bit .31 = saída 32

5.3.3 Opcode 03- Leitura do estado das entradas digitais e saídas:

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

Resposta:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	Long (4)	Input	0x0000000F entradas 1 a 4 atuadas, 5 a 32 não atuadas	Reflete todas as entradas. Bit .0 = entrada 1, Bit .31 = entrada 32

2	Long (4)	Output	0x0000000F saídas 1 a 4 ligadas, 5 a 32 desligadas	Reflete todas as saídas. Bit .0 = saída 1, Bit .31 = saída 32

5.3.4 Opcode 04- Configuração das entradas digitais

Requisição:

Campo	Tamanho(bytes)	Descrição	Exemplo	Observação
1	Long (4)	Endereço	1	Específica o endereço da Entrada Digital
2	Byte (1)	Enabled	1	Habilitação da Entrada 0 = desabilitada, 1 = habilitada
3	Byte (1)	vago	0	Para uso futuro
4	Long (4)	On state delay	20 (Delay=0,2s)	Específica delay para atuação da entrada. Unidade de tempo = 10ms (0,01s)
5	Long (4)	Off state delay	10 (Delay=0,1s)	Específica delay para desatuação da entrada. Unidade de tempo = 10ms (0,01s)
6	Array (33)	vago	0	Para uso futuro (tamanho = 33bytes)

Resposta:
ACK / NACK

5.3.5 Opcode 05- Leitura da configuração das entradas digitais

Requisição:

Campo	Tamanho(bytes)	Descrição	Exemplo	Observação
1	Long (4)	Endereço	1	Específica o endereço da Entrada Digital

Resposta:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	Long (4)	Endereço	1	Endereço da Entrada Digital
2	Byte (1)	Enabled	1	Habilitação da Entrada 0 = desabilitada, 1 = habilitada

3	**Byte (1)**	**vago**	0	Para uso futuro
4	**Long (4)**	**On state delay**	20 (Delay=0,2s)	Específica delay para atuação da entrada. Unidade de tempo = 10ms (0,01s)
5	**Long (4)**	**Off state delay**	10 (Delay=0,1s)	Específica delay para desatuação da entrada. Unidade de tempo = 10ms (0,01s)
6	**Array (33)**	**vago**	0	Para uso future (tamanho = 33bytes)

5.3.6 Opcode 06- Leitura do estado das entradas digitais

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

Resposta:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	**Long (4)**	**Input_data**	0x00000005 Inputs atuados: 1 e 3	Máscara binária referente ao estado dos 32 inputs digitais. 0 = input 1 (...) 31= input 32

5.3.7 Opcode 24- Configuração IP

Requisição:

Campo	Tamanho(bytes)	Descrição	Exemplo	Observação
1	**Byte (1)**	**vago**	0x00	Para uso futuro
2	**Long (4)**	**Source_IP**	0xC0A80064	= 192.168.0.100
3	**Long (4)**	**Subnet Mask**	0xFFFFFF00	= 255.255.255.0
4	**Long (4)**	**Gateway IP**	0xC0A80001	= 192.168.0.1
5	**Int (2)**	**Client_port**	4091	Porta que a MIO recebe os opcodes
6	**String (32)**	**Client_name**	porta_cofre	Nome do equipamento

Resposta:
ACK / NACK

5.3.8 Opcode 25- Leitura da configuração IP

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

Resposta:

Campo	Tamanho(bytes)	Descrição	Exemplo	Observação
1	Byte (1)	vago	0x00	Para uso futuro
2	Long (4)	Source_IP	0xC0A80064	= 192.168.0.100
3	Long (4)	Subnet Mask	0xFFFFFF00	= 255.255.255.0
4	Long (4)	Gateway IP	0xC0A80001	= 192.168.0.1
5	Int (2)	Client_port	4091	Porta que a MIO recebe os opcodes
6	String (32)	Client_name	porta_cofre	Nome do equipamento
7	Int (2)	Emac_address_hi	0x0043	Primeiros 2 octetos do MAC
8	Long (4)	Emac_address_lo	0x4241A5B8	Últimos 4 octetos do MAC

5.3.9 Opcode 30- Formato de envio dos eventos das entradas digitais

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	Long (4)	Event Type	1	1 = Entrada digital
2	Byte (1)	Dia		
3	Byte (1)	Mês		
4	Byte (1)	Ano		
5	Byte (1)	Dia da Semana		1 = Domingo, ..., 7 = Sábado
6	Byte (1)	Hora		
7	Byte (1)	Minuto		
8	Byte (1)	Segundo		
9	Long (4)	Time stamp		Base de tempo CPUx10ms (0,01s)
10	Byte (1)	Link status		0 = OFF, 1 = ON
11	Array (36)	Dados do Evento		Vide tabelas abaixo

Dados do Evento:
- Se Event Type = 1, evento Entrada Digital

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	**Byte (1)**	**Address**	0x12	Entrada 18 atuada
2	**Long (4)**	**Inputs_Status**	0x00000800	Estado de todas as Entradas Digitais Exemplo = entrada 18 atuada
3	**Long (4)**	**Output_Status**	0x00000000	Estado de todas as Saídas Exemplo = nenhuma saída atuada

ATENÇÃO:
- Opcode 30 é enviado da MIOV3 para algum Socket (IP/Porta), ou seja, a MIOV3 não recebe esse opcode.

5.3.10 Opcode 34- Ajuste do relógio interno (RTC)

Requisição:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	**Byte (1)**	**Dia**		
2	**Byte (1)**	**Mes**		
3	**Byte (1)**	**Ano**		
4	**Byte (1)**	**Dia da Semana**		1 = Domingo, ..., 7 = Sábado
5	**Byte (1)**	**Hora**		
6	**Byte (1)**	**Minuto**		
7	**Byte (1)**	**Segundo**		

Resposta:
ACK / NACK

5.3.11 Opcode 35- Leitura do relógio interno (RTC)

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

Resposta:

Campo	Tamanho	Descrição	Exemplo	Observação

	(bytes)			
1	Byte (1)	Dia		
2	Byte (1)	Mes		
3	Byte (1)	Ano		
4	Byte (1)	Dia da Semana		1 = Domingo, ..., 7 = Sábado
5	Byte (1)	Hora		
6	Byte (1)	Minuto		
7	Byte (1)	Segundo		

5.3.12 Opcode 36- Comando de Reset do dispositivo

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

ATENÇÃO:

- A placa da MIOV3 vai resetar 5s após a recepção desse comando.

Resposta:
ACK / NACK

5.3.13 Opcode 37- Comando retorno as configurações de fábrica

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

ATENÇÃO:

- A placa da MIOV3 vai resetar 5s após a recepção desse comando.

Resposta:
ACK / NACK

5.3.14 Opcode 520- Leitura da versão do firmware

Requisição:
Esta requisição não utiliza o campo de dados, apenas o header.

Resposta:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	Int (2)	Fw Version	314	= firmware versão 3.14

5.3.15 Opcode 7000 – Configura tempo que Socket permanece aberto

Requisição:

Campo	Tamanho (bytes)	Descrição	Exemplo	Observação
1	Long (4)	Socket TTL value	10	Específica tempo que o socket permanece aberto Unidade de tempo = 1s

ATENÇÃO:

- A cada recepção de um opcode pela MIOV3 esse tempo configurado é recarregado.
- Se o socket permanecer o tempo configurado sem atividade (sem que a MIOV3 receba opcodes), então a MIOV3 vai fechar esse socket.
- Para manter o socket aberto envie qualquer opcode periodicamente, com um intervalo de tempo menor que o configurado.

Resposta:
ACK / NACK

5.3.16 Opcode 7002- Habilita envio de eventos das entradas digitais

Requisição:

Campo	Tamanho(bytes)	Descrição	Exemplo	Observação
1	Byte (1)	Enabled	1	Habilitação da Entrada 0 = desabilitada, 1 = habilitada

ATENÇÃO:

- Quando a MIO receber esse opcode com o campo “Enable” igual a 1, então ela vai guardar as informações desse Socket (IP e Porta). Quando houver uma alteração que qualquer de suas Entradas Digitais, então a MIO vai enviar um evento (opcode 30) por esse Socket com as informações dessa alteração.
- Se esse Socket fechar, então automaticamente essa função é desabilitada.
- Normalmente essa função é usada em conjunto com o opcode 7000 da seguinte forma:
  - Envia o opcode 7000 com um tempo relativamente alto, como por exemplo 20s, e não feche esse socket;

- Logo em seguida, pelo mesmo socket usado acima, envie o opcode 7002, habilitando o envio de eventos e não feche o socket;
- Antes que o tempo configurado no opcode 7000 acabe, pelo mesmo socket acima, envie qualquer opcode para a MIOV3 e não feche o socket. Por exemplo, a cada 15s envie o opcode 03 (Leitura do estado das entradas digitais e saídas);
- Sempre que uma Entrada Digital for alterada, um opcode 30 será enviado pelo MIOV3 para esse Socket que já está aberto;

Resposta:
ACK / NACK

Contato

Comercial

Matriz

Porto Alegre/RS

Rua Cel. Armando Assis, 222

Brasil - CEP: 91330-010

+55 (51) 3026-2300

Filial

São Caetano do Sul/SP

Alameda Terracota, 215/419

Brasil - CEP: 09531-190

+55 (11) 4063-4300

Suporte Técnico

suporte@commbox.com.br

+55 (51) 3026-2300