Protocolo de Controle de Baixo Nível

Versão 0.7
6 de junho de 2013

Bruno Martins
Grupo de Controle

Histórico de Revisões

Rev	Data	Mudanças
0.7	06/06/2013	• Remoção dos comandos de Escrita e Leitura sincronizadas. • Remoção da checagem de ajustes de valor nos comandos de Escrita • Criação do comando Recalcular Checksum de Curva
0.6	26/04/2013	• Mudança do nome do documento: de “Especificação do Protocolo da Rede Serial de Controle” para “Protocolo de Controle de Baixo Nível” • Reestruturação de todo o conteúdo de modo a dividir a especificação em duas partes: especificação do meio físico e do formato dos pacotes em uma rede serial e especificação dos possíveis conteúdos dos pacotes. • Alteração no comportamento do comando Dados de Post-Mortem. • Definição de Curva e dos comandos de consulta e transferência de Curvas. • Remoção dos comandos de Post-Mortem em favor dos comandos mais genéricos de Curvas. • Mudança no significado do campo TAMANHO do cabeçalho do Protocolo. • Criação do comando Valor Inválido. • Criação do comando Ping.
0.5	18/04/2013	• Mudança do nome do documento: de Especificação do Protocolo PUC para Especificação do Protocolo da Rede Serial de Controle. • Criação do quadro de Histórico de Revisões. • Criação das definições de Mestre e Escravo. • Definição de comandos obrigatórios. • Modificação do comando Lista de Grupos de Variáveis para se tornar semanticamente semelhante ao comando Lista de Variáveis. • Modificação do comando Ler um Grupo de Variáveis para se tornar semanticamente semelhante ao comando Ler uma Variável. • Modificação do comando Escrever em uma Variável e do comando Escrever em um Grupo de Variáveis para que a opção de Postergar Escrita ocupe apenas um *bit*, e não um *byte* inteiro. • Modificação do comando Grupo de Variáveis para não mais retornar se cada Variável é de escrita ou leitura (informação redundante). • Modificação nas respostas de comandos que deveriam ser usados apenas em *Multicast.* • Introdução do comando Erro Interno. • Modificação do comando Grupo Criado, para que passe a indicar se o grupo criado pode ser escrito *(bit* mais significativo). • Remoção dos comandos Variável Somente Leitura e Grupo contém Variáveis de Leitura. • Criação do comando Somente Leitura, substituindo os dois removidos acima.
0.4	25/03/2013	• Criação do comando Grupo de Variáveis, para ser utilizado como resposta ao comando Criar Grupo de Variáveis.

		• Modificação do comando de Remover de Grupo de Variável para remover todos os Grupos. • Modificação do comando de Desinscrever de Grupo de Multicast para se desinscrever de todos os Grupos.
0.3	22/03/2013	• Explicitação da faixa de valores possíveis de serem utilizados nos campos ID e TAMANHO das Variáveis. • Criação do campo TIPO para Grupos de Variáveis. • Introdução dos Grupos de *Multicast* e seus comandos associados. • Introdução dos comandos de Leitura Sincronizada. • Relocação dos comandos Consultar Grupo de Variáveis e Grupo de Variáveis para a seção Comandos de Consulta. • Modificação da seção Comandos de Manipulação de Grupos de Variáveis para somente Comandos de Manipulação de Grupos. • Modificação dos comandos de Escrita para acomodar o caso de Escrita Sincronizada. • Remoção dos comandos exclusivos de Escrita Sincronizada.
0.2	18/03/2013	• Criação dos comandos de Escrita Sincronizada. • Criação de cinco novos códigos de erro.
0.1	14/03/2013	• Versão Inicial.

Sumário

1 Introdução

A Rede de Controle para o Sirius será construída sobre dois meios de comunicação entre os equipamentos: Ethernet e Serial RS 485. O Grupo de Controle utilizará Ethernet nos níveis mais altos da hierarquia da Rede de Controle; os equipamentos de atuação e medição, no nível mais baixo da hierarquia, serão interconectados através da rede Serial.

Este documento descreve a comunicação entre os dispositivos que compõem o chamado "Baixo Nível" do Sistema de Controle. Este nível compreende todos os dispositivos que controlam diretamente um equipamento ligado ao acelerador: BPM, fonte de corrente, sensor de temperatura, etc. A descrição desta comunicação se divide em duas partes.

A primeira parte dedica-se a especificar a transmissão de um pacote no meio físico de comunicação das redes Seriais RS 485, bem como o formato dos pacotes que circularão neste meio. Esta especificação deve ser seguida por todos os equipamentos que se conectem a uma rede serial com um *Single Board Computer* – SBC – do Grupo CON como mestre.

A segunda parte especifica as mensagens do Protocolo de Controle de Baixo Nível, detalhando seus formatos e funções.

Este protocolo foi desenvolvido com a PUC em mente. Entretanto, ele foi projetado de modo a permitir sua utilização por outros *hardwares* de outros grupos do LNLS. A PUC – Placa Universal de Controle – é um *hardware* desenvolvido pelo Grupo de Controle com o propósito de controlar e adquirir dados de equipamentos do acelerador através de entradas e saídas analógicas e digitais, formando a camada mais baixa do Sistema de Controle. Grupos que desejem implementar seu próprio *hardware* de aquisição e controle podem utilizar a PUC como referência de implementação.

2 Comunicação Serial

Na rede Serial RS 485 todos os dados de um pacote devem ser transmitidos em binário, ou seja, nenhum valor de *byte* deve ter significado especial. O fim de pacote deve ser indicado por um silêncio na linha serial de duração equivalente à duração de dois *bytes.* Por exemplo, em uma rede serial 10 Mbps um silêncio de 1.6 µs após a transmissão de um pacote marca seu fim. Se o Mestre não obter resposta de um Nó em 1 ms o pacote deve ser dado como perdido e reenviado.

Os pacotes que circulam na rede serial **devem** ter seu formato bem definido. **Devem** existir dois *bytes* para endereçamento, nesta ordem: DESTINO e ORIGEM. O *byte* de ORIGEM **deve** indicar o endereço do transmissor da mensagem. O *byte* DESTINO, entretanto, **pode** indicar tanto um dispositivo ou um grupo de dispositivos aos quais se endereça a mensagem. **Deve** existir, também, um *byte* ao final da mensagem, contendo seu CHECKSUM, como apresentado na Tabela 1. O CKECKSUM **deve** conter a soma de todos os *bytes* anteriores, em complemento de 2, de modo que a soma de todos os *bytes* do pacote seja nula. Cada pacote tem a função de transmitir uma Mensagem, que deve ser interpretada e executada de acordo com o Protocolo de Controle de Baixo Nível.

Endereçamento		**Mensagem**	**Checksum**
DESTINO	ORIGEM		CHECKSUM

Tabela 1- Estrutura de um pacote da Rede Serial

2.1 Endereçamento

Os dispositivos em uma rede serial devem ser endereçados por um número simples entre 0 (zero) e 31, inclusive. Esta faixa restringe a existência de apenas trinta e dois dispositivos por rede serial. O dispositivo de endereço 0 (zero) é denominado Mestre da rede Serial, sendo os outros dispositivos chamados Nós ou Escravos. Toda rede deve ser composta por **exatamente** um mestre e **no mínimo** um Nó. Os Nós de uma rede Serial devem receber e interpretar pacotes enviados diretamente a ele ou ao um Grupo de *Multicast* do qual ele faça parte. Cada Nó deve responder **apenas** a pacotes enviados diretamente a ele, ou seja, pacotes para Grupos de *Multicast* **não devem** ser respondidos.

2.1.1 Grupos de *Multicast*

Dispositivos podem ser agrupados nos chamados Grupos de *Multicast.* Os Grupos de *Multicast* podem assumir endereços entre 240 e 255, constituindo dezesseis possíveis Grupos. O Grupo de endereço 255 é um Grupo especial denominado *Broadcast*. Todos os dispositivos em uma rede serial **devem** pertencer ao Grupo de *Broadcast*. Um Nó pode pertencer a mais de um Grupo de *Multicast*. Reforça-se que pacotes enviados a Grupos de *Multicast* **não devem** ser respondidos pelos destinatários.

As faixas de endereços em uma Rede Serial estão especificadas na Tabela 2.

Endereço	**Usado para**
0	Mestre da Rede
1 a 31	Identificação de Nó
32 a 239	RESERVADO
240 a 254	Grupo de *Multicast*
255	*Broadcast*

Tabela 2- Endereços na Rede Serial

3 Protocolo de Controle de Baixo Nível

Nesta seção especificam-se termos e conceitos que serão usados no restante deste documento.

3.1 Rede, Mensagem, Comando, Mestre e Nó (ou Escravo)

Os dispositivos conectados pelo Protocolo de Controle de Baixo Nível constituem uma **Rede**. Os componentes da **Rede** se comunicam através da troca de **Mensagens**. Cada **Mensagem** contém um **Comando**, que pode ser tanto um pedido de execução de uma ação quanto uma resposta à tal execução.

Assume-se que os dispositivos da **Rede** atuam em um de dois possíveis papéis: **Mestre** ou **Nó** (também chamado de **Escravo**). Deve existir **exatamente** um **Mestre** por **Rede**. A quantidade de **Nós** por **Rede** não é limitada pelo Protocolo de Controle de Baixo Nível.

3.2 Variável

A Variável é a entidade central do Protocolo. Cada Nó da rede é responsável por gerenciar um número determinado de Variáveis. Cada Variável corresponde a um valor independente que pode ser lido e, em alguns casos, também escrito. Cada Variável deve ter 4 informações associadas a ela, de acordo com a Tabela 3. É importante ressaltar que uma Variável de escrita **deve** poder ser lida. Por exemplo, caso se leia o valor de uma Variável correspondente a um conversor D/A, o valor retornado é o último valor de ajuste recebido. Já a escrita em uma Variável de leitura **não deve** ser permitida. Os ID's das Variáveis devem ser **contínuos** e começar em 0, ou seja, se houver 4 Variáveis em um Nó, por exemplo, seus ID's **devem** ser, **necessariamente**, 0, 1, 2 e 3.

Informação	Descrição
ID	Número de 0 a 127 que identifica univocamente a Variável dentro do Nó
TIPO	Indica se a Variável é de leitura (0) ou de escrita (1)
TAMANHO	Indica quantos *bytes* são necessários para armazenar o valor da Variável, de 1 a 127
VALOR	Armazena o valor da Variável

Tabela 3- Informações associadas a cada Variável

Estas informações são obtidas pelo Mestre da rede através de comandos específicos de consulta (Comandos de Consulta (0x0_)).

3.3 Grupo de Variáveis

É possível que sejam definidos Grupos de Variáveis para que certos conjuntos de Variáveis possam ser lidos e/ou escritos com um único comando. Cada Grupo de Variáveis **deve** conter três informações: um número de identificação (ID), seu tipo (0: contém variáveis de leitura e escrita, 1: contém somente variáveis de escrita) e uma lista com os ID's das Variáveis pertencentes àquele Grupo. **Devem** existir, no mínimo, três Grupos de Variáveis, descritos na Tabela 4, chamados Grupos Padrão. O ID de um Grupo pode assumir valores entre 0 e 127, inclusive. Os Grupos Padrão **não devem** ser modificados ou excluídos. Sua existência com os ID's indicados **deve** ser garantida.

ID	TIPO	Descrição do grupo
0	0	Todas as Variáveis
1	0	Variáveis de Leitura
2	1	Variáveis de Escrita

Tabela 4- Grupos Padrão

Além dos Grupos Padrão, novos Grupos podem ser criados e excluídos através dos Comandos de Manipulação de Grupos de Variáveis(0x3_). Os Grupos, assim como as Variáveis, **devem** ter seus ID's contínuos. Ou seja, se houver 5 Grupos em um nó, por exemplo, **necessariamente** seus ID's serão 0, 1, 2, 3, e 4.

3.4 Curva

Curva é o nome dado a uma sequência grande de *bytes*, que podem ou não estar relacionados ente si. Os valores de Curva podem ser transmitidos tanto do Mestre para o Nó quanto do Nó para o Mestre. As Curvas armazenadas em um Nó **devem** ter cinco informações associadas, descritas na Tabela 36.

Informação	Descrição
ID	Número de 0 a 127 que identifica univocamente a Curva dentro do Nó
TIPO	Indica se a Curva é de leitura (0) ou de escrita (1)
TAMANHO	Indica o número de blocos contidos na Curva, menos 1.
VALORES	Armazena os valores da Curva
CHECKSUM	Número de 16 *bytes* que representa o *hash* MD5 de todos os VALORES da Curva

Tabela 5- Informações associadas a cada Curva

Os ID's das Curvas **devem** ser contínuos e começar em 0, ou seja, se houver 4 Curvas em um Nó, por exemplo, seus ID's **devem** ser, necessariamente, 0, 1, 2 e 3. O TIPO **deve** indicar se os

valores da Curva podem ser escritos (1) (curvas de Ciclagem ou Rampa, por exemplo) ou apenas lidos (0) (curva de Post-Mortem, por exemplo). Cada Curva **deve** ser limitada a um tamanho de 256 blocos, sendo cada bloco de tamanho 16384 bytes, totalizando um máximo de 4MB por Curva. O campo TAMANHO **deve** armazenar o número de blocos da Curva, subtraindo 1. Cada Curva no Nó **pode** (não é obrigatório) possuir um valor de CHECKSUM associado a ela. Este CHECKSUM **deve** ser calculado através da função *hash* MD5. Caso a Curva não possua CHECKSUM, seu valor **deve** ser 0 (zero).

3.5 Tipo de Protocolo

O Protocolo aqui descrito é um protocolo do tipo *token* (ou de ficha). Só tem permissão de transmissão pela Rede aquele que possui a ficha. No caso deste Protocolo a ficha é implícita. Assim, toda comunicação é iniciada pelo Mestre. Uma vez que o Mestre envia uma mensagem diretamente a um dos Escravos, fica implícito que este Escravo possui a ficha até enviar uma resposta, momento em que a ficha retorna ao Mestre. O Protocolo não armazena estado, ou seja, cada par comando/resposta representa uma única transação independente.

3.6 Formato da Mensagem

Uma mensagem do Protocolo deve ter, no mínimo, dois *bytes*, constituintes de seu cabeçalho: COMANDO e TAMANHO. O campo COMANDO especifica qual o comando que deve ser executado pelo Nó. Os códigos de comandos existentes no Protocolo estão descritos na seção Comandos do Protocolo de Baixo Nível. O campo TAMANHO indica quantos *bytes* estão contidos na Carga Útil do pacote. Caso o comando não contenha carga útil, o campo TAMANHO deve conter o valor 0 (zero). A estrutura da mensagem está ilustrada na Tabela 6.

Cabeçalho		**Carga Útil**								
COMANDO	TAMANHO									

Tabela 6- Estrutura de uma Mensagem do Protocolo de Controle de Baixo Nível

O campo TAMANHO deve ser interpretado como dois campos de *bits*: o *bit* mais significativo é chamado de *m* (de “modo”) e os sete *bits* menos significativos são chamados coletivamente de *n* (de “número”). Esse arranjo está ilustrado na Tabela 7.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Nome	m	n						

Tabela 7- Divisão do campo TAMANHO

O valor de *m* indica como o valor em *n* deve ser interpretado. O valor do campo TAMANHO é calculado segundo a Equação 1.

$$\begin{cases} \text{Se } m = 0, \text{TAMANHO} = n \\ \text{Se } m = 1, \text{TAMANHO} = 128(n+1) + 2 \end{cases}$$

Equação 1- Interpretação do campo TAMANHO

Uma justificativa para a escolha deste arranjo encontra-se na seção Racionalização do campo TAMANHO do Protocolo. A presença da soma “+2” se justifica pelo comando Bloco de Curva (0x41) que precisa de dois *bytes* de informação para identificar o bloco a que se refere.

Caso se deseje enviar uma Mensagem com mais de 128 *bytes* de Carga, mas que não seja um múltiplo de 128, **deve-se** preencher a Carga com *bytes* nulos até que a Carga tenha um tamanho múltiplo de 128.

Desta maneira, é possível enviar comandos de TAMANHO entre 0 e 127 com incrementos de 1 *byte* e entre 130 e 16386 com incrementos de 128 *bytes.*

3.7 Configuração Hipotética

Nos exemplos de comandos considera-se a presença de uma PUC – Placa Universal de Controle - na Rede. A PUC foi projetada para ser flexível, admitindo diferentes configurações, no que tange o número de conversores A/D, D/A e entradas e saídas digitais. Assim, considera-se em todos os exemplos da seção Comandos do Protocolo de Baixo Nível uma PUC com 4 conversores A/D, 4 conversores D/A, 1 *byte* de saídas digitais e 1 *byte* de entradas digitais. Assim, a PUC é responsável por 10 Variáveis: uma para cada conversor A/D, uma para cada conversor D/A e uma para cada *byte* de entrada e saída digital.

As PUC's atribuem ID's mais baixos aos conversores A/D, depois aos conversores D/A, seguidos das entradas digitais e, por fim, das saídas digitais. Deste modo, para a PUC hipotética, as 10 Variáveis teriam suas informações como na Tabela 8. Os conversores da PUC têm precisão de 18 *bits*, sendo necessários, no mínimo, 3 *bytes* de espaço para armazenar seu VALOR. Os Grupos de Variáveis presentes na PUC, após a inicialização, estão apresentados na Tabela 9.

Na inicialização a PUC participa de apenas um Grupo serial de *Multicast*: o de *broadcast*.

Variável	ID	TIPO	TAMANHO
Primeiro ADC	0	0	3
Segundo ADC	1	0	3
Terceiro ADC	2	0	3
Quarto ADC	3	0	3
Primeiro DAC	4	1	3
Segundo DAC	5	1	3
Terceiro DAC	6	1	3
Quarto DAC	7	1	3
Entrada Digital	8	0	1
Saída Digital	9	1	1

Tabela 8- Variáveis na PUC hipotética

ID	TIPO	Descrição do grupo	Lista de Variáveis na PUC de exemplo
0	0	Todas as Variáveis	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
1	0	Variáveis de Leitura	0, 1, 2, 3, 8
2	1	Variáveis de Escrita	4, 5, 6, 7, 9

Tabela 9- Grupos de Variáveis padrão na PUC hipotética

4 Comandos do Protocolo de Baixo Nível

Os códigos aceitos no campo COMANDO das mensagens estão descritos nesta seção. Estes comandos estão divididos em classes de comandos, sendo agrupados pela sua semelhança semântica. Cada comando consiste de um *byte*, sendo seu *nibble* mais significativo indicativo de sua classe. Em geral, segue-se a convenção de que comandos pares são enviados pelo Mestre para um Escravo e comandos ímpares são enviados de um Escravo para o Mestre, existindo exceções (por exemplo, nos casos de códigos de erro (seção Comandos de Erro (0xE_)). Se o Nó receber um comando com um formato inesperado, ou seja, com o número de *bytes* indicado no campo TAMANHO diferente do número de *bytes* de fato enviados no corpo da mensagem, **deve-se** retornar o comando Mensagem Mal Formada (0xE1). Existe um conjunto de comandos cuja implementação deve ser obrigatória. Estes comandos estão devidamente identificados através da informação "OBRIGATÓRIO?", na seção respectiva de cada comando. Caso um comando não obrigatório enviado a um Nó não tenha sido implementado nele, **deve-se** responder com o comando Operação Não Suportada (0xE2). Caso o comando possua Carga Útil, é apresentado um exemplo de sua utilização. Caso o comando não possua Carga Útil, seu pacote se resume a dois *bytes*: o primeiro *byte* contendo seu código de comando e o segundo *byte* contendo seu tamanho, 0 (zero).

4.1 Comandos de Consulta (0x0_)

Os comandos de consulta são utilizados para se obter informações sobre os parâmetros de operação de um Nó.

4.1.1 Consultar *Status* (0x00)

COMANDO: 0x00
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Pedido para que o Nó retorne informações sobre seu *status* de operação
SENTIDO: Mestre → Nó

A resposta a esse comando **deve** ser o comando Status (0x01).

4.1.2 *Status* (0x01)

COMANDO: 0x01
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Informações sobre o *status* de operação do remetente
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: A DEFINIR
ESTRUTURA DA CARGA: A DEFINIR
EXEMPLO: A DEFINIR

Ainda deve-se definir quais informações de *status* devem ser retornadas.

4.1.3 Consultar Lista de Variáveis (0x02)

COMANDO: 0x02
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Pedido para que o Nó retorne sua lista de Variáveis
SENTIDO: Mestre → Nó

A resposta a esse comando **deve** ser o comando Lista de Variáveis (0x03).

4.1.4 Lista de Variáveis (0x03)

COMANDO: 0x03
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Lista de Variáveis controladas pelo nó
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: (número de Variáveis)
ESTRUTURA DA CARGA: Tabela 10
EXEMPLO: Tabela 11

As Variáveis **devem** ser retornadas na Carga da Mensagem, uma a uma, na sua ordem de ID. **Não deve** existir descontinuidade nos ID's das Variáveis. A primeira Variável é considerada com valor de ID sendo 0. Para cada Variável, **deve-se** retornar um *byte* de informação. O *bit* mais significativo deste *byte* **deve** indicar se a Variável é de leitura (*bit* = 0) ou escrita (*bit* = 1). Os sete *bits* restantes **devem** conter o tamanho do VALOR da Variável. Assim, pode-se notar, no exemplo apresentado na Tabela 11, que os quatro primeiros *bytes* da Carga da Mensagem indicam quatro Variáveis de leitura de tamanho 3. Já os quatro *bytes* seguintes indicam quatro Variáveis de escrita de tamanho 3. Por fim, os dois últimos *bytes* indicam uma Variável de leitura e uma Variável de escrita, ambas de tamanho 1.

Carga		
Tipo (1 *bit*) | Tamanho da primeira Variável (7 *bits*)	...	Tipo (1 *bit*) | Tamanho da última Variável (7 *bits*)

Tabela 10- Estrutura do comando Lista de Variáveis

Cabeçalho		**Carga**									
03	0A	03	03	03	03	83	83	83	83	01	81

Tabela 11- Exemplo do comando Lista de Variáveis

4.1.5 Consultar Lista de Grupos de Variáveis (0x04)

COMANDO: 0x04
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Pedido da lista de Grupos de Variáveis disponíveis no escravo
SENTIDO: Mestre → Nó

A resposta a esse comando **deve** ser o comando Lista de Grupos de Variáveis (0x05).

4.1.6 Lista de Grupos de Variáveis (0x05)

COMANDO: 0x05
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Lista de Grupos de Variáveis no escravo
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: (número de Grupos de Variáveis)
ESTRUTURA DA CARGA: Tabela 12
EXEMPLO: Tabela 13

Os Grupos de Variáveis do nó **devem** ser retornados na Carga da Mensagem, um a um, na sua ordem de ID. **Não deve** existir descontinuidade nos ID's dos Grupos. O primeiro Grupo é considerado com valor de ID sendo 0. Para cada Grupo, **deve-se** retornar um *byte* de informação. O *bit* mais significativo deste *byte* **deve** indicar se o Grupo é de leitura (*bit* = 0) ou escrita (*bit* = 1). Os sete *bits* restantes **devem** conter a quantidade de Variáveis no Grupo. O exemplo lista os três Grupos padrão em um nó. O primeiro grupo é de leitura (*bit* mais significativo 0) e contém dez Variáveis. O segundo também é de leitura e contém cinco Variáveis. Por fim, o último Grupo é de escrita (*bit* mais significativo 1) e contém cinco Variáveis.

Carga		
Tipo (1 *bit*) | Número de Variáveis no primeiro Grupo (7 *bits*)	…	Tipo (1 *bit*) | Número de Variáveis no último Grupo (7 *bits*)

Tabela 12- Estrutura do comando Lista de Grupos de Variáveis

Cabeçalho		**Carga**		
05	03	0A	05	85

Tabela 13- Exemplo de comando Lista de Grupos de Variáveis

4.1.7 Consultar Grupo de Variáveis (0x06)

COMANDO: 0x06
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Consulta para se obter a lista de Variáveis pertencentes a um Grupo
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 1
ESTRUTURA DA CARGA: Tabela 14
EXEMPLO: Tabela 15

- **Deve-se** responder a esse comando com o comando Grupo de Variáveis (0x07).
- Caso o ID do Grupo não exista, **deve-se** responder com o comando ID Inválido (0xE3).
- Caso o tamanho da Carga seja diferente de 1, **deve-se** retornar o comando Tamanho da Carga Inválido (0xE5).

O exemplo faz uma consulta ao Grupo de ID 2.

Carga
ID do Grupo

Tabela 14- Estrutura do comando Consultar Grupo de Variáveis

Cabeçalho		**Carga**
06	01	02

Tabela 15- Exemplo de comando Consultar Grupo de Variáveis

4.1.8 Grupo de Variáveis (0x07)

COMANDO: 0x07
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Lista de Variáveis que compõem um Grupo
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: (quantidade de Variáveis no Grupo consultado)
ESTRUTURA DA CARGA: Tabela 16
EXEMPLO: Tabela 17

As Variáveis contidas no Grupo de Variáveis consultado **devem** ser retornados na Carga da Mensagem, uma a uma, na sua ordem de ID. O exemplo consiste na resposta à consulta ao Grupo de Variáveis de escrita (ID 2). Esta resposta inclui os ID's dos quatro conversores D/A e da saída digital.

Carga		
ID da primeira Variável no Grupo	…	ID da última Variável no Grupo

Tabela 16- Estrutura do comando Grupo de Variáveis

Cabeçalho		Carga				
07	05	04	05	06	07	09

Tabela 17- Exemplo de comando Grupo de Variáveis

4.1.9 Consultar Lista de Curvas (0x08)

COMANDO: 0x08
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Pedido para que o Nó retorne sua lista de Curvas
SENTIDO: Mestre → Nó

A resposta a esse comando **deve** ser o comando Lista de Curvas (0x09).

4.1.10 Lista de Curvas (0x09)

COMANDO: 0x09
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Lista de Curvas no Nó
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: 18*(número de Curvas)
ESTRUTURA DA CARGA: Tabela 18
EXEMPLO: Tabela 19

As Curvas **devem** ser retornadas na Carga da Mensagem, uma a uma, na sua ordem de ID. **Não deve** existir descontinuidade nos ID's das Curvas. A primeira Curva é considerada com valor de ID sendo 0. Para cada Curvas, **deve-se** retornar 18 *bytes* de informação. O primeiro *byte* **deve** indicar se a Curva é de leitura (*byte* = 0) ou escrita (*byte* = 1). O segundo *byte* **deve** conter o número de blocos da Curva, menos 1. Os dezesseis *bytes* restantes **devem** conter o Checksum da Curva (*byte* mais significativo primeiro) ou o valor 0 (zero) caso a Curva não possua Checksum.

No exemplo apresentado na Tabela 19 mostra-se a resposta de um Nó que contenha uma Curva de leitura, de tamanho 32 blocos e que não possui Checksum.

Carga										
Tipo (1ª Curva)	Blocos-1 (1ª Curva)	Checksum (mais sig.) (1ª Curva)	...	Checksum (menos sig.) (1ª Curva)	...	Tipo (ult. Curva)	Blocos-1 (ult. Curva)	Checksum (mais sig.) (ult. Curva)	...	Checksum (menos sig.) (primeira Curva)

Tabela 18- Estrutura do comando Lista de Curvas

Cabeçalho		**Carga**																	
09	12	00	31	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Tabela 19- Exemplo do comando Lista de Curvas

4.2 Comandos de Leitura (0x1_)

4.2.1 Ler Variável (0x10)

COMANDO: 0x10
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Leitura do valor de uma Variável
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 1
ESTRUTURA DA CARGA: Tabela 20
EXEMPLO: Tabela 21

Este comando faz o pedido de leitura do VALOR de uma Variável. O *bit* mais significativo do campo ID da Variável é reservado e **deve** ser 0 (zero).

- **Deve-se** responder a esse comando com o comando Leitura de uma Variável (0x11).
- Caso não exista uma variável com o ID passado, **deve-se** retornar o comando ID Inválido (0xE3).
- Caso o tamanho da Carga seja diferente de 1, **deve-se** retornar o comando Tamanho da Carga Inválido (0xE5).

O exemplo faz a leitura da Variável de ID 3 (último conversor A/D).

Carga
ID da Variável

Tabela 20- Estrutura do comando Ler Variável

Cabeçalho		**Carga**
10	01	03

Tabela 21- Exemplo de comando Ler Variável

4.2.2 Leitura de uma Variável (0x11)

COMANDO: 0x11
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Valor de uma Variável
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: (tamanho do campo VALOR da Variável)
ESTRUTURA DA CARGA: Tabela 22
EXEMPLO: Tabela 23

O exemplo consiste na resposta à leitura do VALOR da Variável com ID 3, considerando que este valor seja o valor máximo alcançado pelo conversor A/D (todos os 18 bits altos).

Carga		
Primeiro *byte* (mais significativo) do VALOR da Variável	…	Último *byte* (menos significativo) do VALOR da Variável

Tabela 22- Estrutura do comando Leitura de uma Variável

Cabeçalho		**Carga**		
11	03	03	FF	FF

Tabela 23- Exemplo de comando Leitura de uma Variável

4.2.3 Ler um Grupo de Variáveis (0x12)

COMANDO: 0x12
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Leitura dos valores de um Grupo de Variáveis
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 1
ESTRUTURA DA CARGA: Tabela 24
EXEMPLO: Tabela 25

Este comando faz o pedido de leitura do campo VALOR das Variáveis em um Grupo. O *bit* mais significativo do campo ID do Grupo é reservado e **deve** ser 0 (zero).

- **Deve-se** responder a esse comando com o comando Leitura de um Grupo de Variáveis (0x13).
- Caso não exista um Grupo com o ID passado, **deve-se** retornar o comando ID Inválido (0xE3).
- Caso o tamanho da Carga seja diferente de 1, **deve-se** retornar o comando Tamanho da Carga Inválido (0xE5).

O exemplo faz a leitura do Grupo de Variáveis de ID 1 (todas as Variáveis de leitura).

Carga
ID do Grupo

Tabela 24- Estrutura do comando Ler Grupo de Variáveis

Cabeçalho		**Carga**
12	01	01

Tabela 25- Exemplo de comando Ler Grupo de Variáveis

4.2.4 Leitura de um Grupo de Variáveis (0x13)

COMANDO: 0x13
DESCRIÇÃO: Contém os valores das Variáveis de um Grupo de Variáveis
OBRIGATÓRIO?: Sim
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: (soma dos tamanhos dos campos VALOR das Variáveis)
ESTRUTURA DA CARGA: Tabela 26
EXEMPLO: Tabela 27

Deve-se retornar o VALOR de cada Variável na ordem crescente de ID das Variáveis. O exemplo consiste na resposta à leitura dos valores de todas as Variáveis de leitura (Grupo de ID 1), considerando que os conversores A/D estejam com seu valor máximo (todos os 18 bits altos) e que a entrada digital tenha o valor 0xAA.

Carga						
Primeiro *byte* da primeira Variável	…	Último *byte* da primeira Variável	...	Primeiro *byte* da última Variável	…	Último *byte* da última Variável

Tabela 26- Estrutura do comando Leitura de um Grupo de Variáveis

Cabeçalho		**Carga**												
13	0C	03	FF	FF	03	FF	FF	03	FF	FF	03	FF	FF	AA

Tabela 27- Exemplo de comando Leitura de um Grupo de Variáveis

4.3 Comandos de Escrita (0x2_)

4.3.1 Escrever em uma Variável (0x20)

COMANDO: 0x20
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Escrita no valor de uma Variável
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 1 + (tamanho do VALOR da Variável)
ESTRUTURA DA CARGA: Tabela 28
EXEMPLO: Tabela 29

O comando de escrita de uma Variável **deve** ser enviado **apenas** para Variáveis de escrita. O *bit* mais significativo do campo ID da Variável é reservado e **deve** ser 0 (zero). Caso algum erro aconteça, a escrita **deve** ser ignorada pelo Nó. O comando a ser retornado depende de alguns fatores:

- Caso o comando tenha sido executado com sucesso, **deve-se** retornar o comando OK (0xE0);
- Caso a ID da Variável não exista, **deve-se** retornar o comando ID Inválido (0xE3);
- Caso o o tamanho da Carga seja diferente do tamanho esperado, **deve-se** responder com o comando Tamanho da Carga Inválido (0xE5);
- Caso o ID de Variável passado seja de uma Variável de leitura, **deve-se** responder com o comando Somente Leitura (0xE6).

O exemplo ilustra o ajuste imediato do valor do primeiro conversor D/A (ID 4) para 0x1BBBB.

Carga			
ID da Variável	Primeiro *byte* (mais significativo) do VALOR da Variável	…	Último *byte* (menos significativo) do VALOR da Variável

Tabela 28- Estrutura do comando Escrever em uma Variável

Cabeçalho		**Carga**			
20	04	04	01	BB	BB

Tabela 29- Exemplo de comando Escrever em uma Variável

4.3.2 Escrever em Grupo de Variáveis (0x22)

COMANDO: 0x22
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Escreve nos valores das Variáveis de um Grupo de Variáveis
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 2 + (soma dos tamanhos dos campos VALOR das Variáveis)
ESTRUTURA DA CARGA: Tabela 30
EXEMPLO: Tabela 31

O comando de escrita em um Grupo de Variáveis deve ser enviado para Grupos contendo apenas Variáveis de escrita. O *bit* mais significativo do campo ID do Grupo é reservado e **deve** ser 0 (zero). Caso se tente escrever em um Grupo com ao menos uma Variável de leitura, o Nó **deve** ignorar todas as escritas e retornar o código de erro apropriado.

- Caso o comando tenha sido executado com sucesso, **deve-se** retornar o comando OK (0xE0);
- Caso a ID do Grupo de Variável não exista, **deve-se** retornar o comando ID Inválido (0xE3);
- Caso o o tamanho da Carga seja diferente do tamanho esperado, **deve-se** responder com o comando Tamanho da Carga Inválido (0xE5);
- Caso o ID do Grupo de Variáveis se refira a um Grupo com ao menos uma Variável de leitura, **deve-se** responder com o comando Somente Leitura (0xE6);

O exemplo ilustra o ajuste imediato de todos os conversores D/A para 0x1BBBB e da saída digital para 0xCC, usando-se o Grupo de ID 2.

Carga							
ID do Grupo	Primeiro *byte* da primeira Variável	…	Último *byte* da primeira Variável	...	Primeiro *byte* da última Variável	…	Último *byte* da última Variável

Tabela 30- Estrutura do comando Escrever em Grupo de Variáveis

Cabeçalho		**Carga**													
22	0E	02	01	BB	BB	01	BB	BB	01	BB	BB	01	BB	BB	CC

Tabela 31- Exemplo de comando Escrever em Grupo de Variáveis

4.4 Comandos de Manipulação de Grupos de Variáveis(0x3_)

4.4.1 Criar Grupo de Variáveis (0x30)

COMANDO: 0x30
OBRIGATÓRIO?: Não
DESCRIÇÃO: Cria um novo Grupo de Variáveis
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: (número de Variáveis no Grupo)
ESTRUTURA DA CARGA: Tabela 32
EXEMPLO: Tabela 33

Este comando cria um novo Grupo de Variáveis, para ser adicionado aos Grupos já existentes. **Deve-se** usar sete *bits* para o ID do Grupo, permitindo a existência de 127 Grupos de Variáveis. Porém, três desses Grupos são pré-definidos, como descrito na seção Grupo de Variáveis. **Deve-se**, neste comando, especificar uma lista com os ID's das Variáveis que pertencerão a este Grupo. O ID do Grupo é atribuído automaticamente, sendo **necessariamente** igual ao valor do ID do último Grupo criado, somado a 1.

- Caso a criação aconteça com sucesso, **deve-se** retornar o comando Grupo de Variáveis Criado (0x31);
- Caso o número de Variáveis seja nulo ou maior que a quantidade de Variáveis existentes, **deve-se** retornar o comando Tamanho da Carga Inválido (0xE5);
- Caso não haja memória suficiente para a criação do Grupo, **deve-se** retornar o comando Memória Insuficiente (0xE7);

O exemplo ilustra a criação de um Grupo contendo todos os conversores D/A.

Carga		
ID da primeira Variável do Grupo	…	ID da última variável do Grupo

Tabela 32- Estrutura do comando Criar Grupo de Variáveis

Cabeçalho		**Carga**			
30	04	04	05	06	07

Tabela 33- Exemplo de comando Criar Grupo de Variáveis

4.4.2 Grupo de Variáveis Criado (0x31)

COMANDO: 0x31
OBRIGATÓRIO?: Não
DESCRIÇÃO: Confirmação da criação de um Grupo de Variáveis
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: 1
ESTRUTURA DA CARGA: Tabela 34
EXEMPLO: Tabela 35

Este comando indica que a criação do Grupo de Variáveis, feita através do comando Criar Grupo de Variáveis (0x30), aconteceu com sucesso. Sua Carga **deve** conter o TIPO e o ID do Grupo recentemente criado. Caso o Grupo seja de escrita, seu TIPO será 1.

O exemplo ilustra a resposta à criação do Grupo como mostrada no exemplo do comando anterior.

Carga
Tipo do Grupo (1 *bit*) | ID do Grupo (7 *bits*)

Tabela 34- Estrutura do comando Grupo de Variáveis Criado

Cabeçalho		**Carga**
31	01	83

Tabela 35- Exemplo de comando Grupo de Variáveis Criado

4.4.3 Remover Todos os Grupos de Variáveis (0x32)

COMANDO: 0x32
OBRIGATÓRIO?: Não
DESCRIÇÃO: Pedido de remoção de todos os Grupo de Variáveis
SENTIDO: Mestre → Nó

Este comando **deve** remover todos os Grupos de Variáveis criados, **exceto** os grupos padrão, descritos na seção Grupo de Variáveis. **Deve-se** retornar o comando OK (0xE0).

4.5 Comandos de Transferência de Curvas (0x4_)

Nesta seção estão descritos os comandos para transferência de Curva.

4.5.1 Transmitir Bloco de Curva (0x40)

COMANDO: 0x40
OBRIGATÓRIO?: Não
DESCRIÇÃO: Pedido para que um Nó transmita um bloco de Curva
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 2
ESTRUTURA DA CARGA: Tabela 36
EXEMPLO: Tabela 37

Este comando pede ao Nó para que ele transmita um determinado bloco de uma determinada Curva.

- Caso os dados do comando sejam válidos, **deve-se** retornar o comando Bloco de Curva (0x41);
- Caso o ID da Curva seja inválido, **deve-se** retornar o comando ID Inválido (0xE3);
- Caso o *offset* do bloco seja inválido (maior que o número de blocos da Curva), **deve-se** retornar o comando Valor Inválido (0xE4);

O exemplo ilustra o pedido do quinto bloco da Curva de ID 3.

Carga	
ID da Curva	*Offset* do bloco

Tabela 36- Estrutura do comando Transmitir Bloco de Curva

Cabeçalho		**Carga**	
40	02	03	04

Tabela 37- Exemplo de comando Transmitir Bloco de Curva

4.5.2 Bloco de Curva (0x41)

COMANDO: 0x41
OBRIGATÓRIO?: Não
DESCRIÇÃO: Transmissão de um bloco de uma Curva
SENTIDO: Mestre → Nó ou Nó → Mestre
TAMANHO DA CARGA: 16386
ESTRUTURA DA CARGA: Tabela 38
EXEMPLO: Tabela 39

Transmissão de um bloco de Curva tanto pelo Nó quanto pelo Mestre. Se a transmissão for feita do Mestre para o Nó, entende-se como uma escrita no bloco indicado. Neste caso, o Checksum da Curva especificada é **zerado**. O cálculo do Checksum deve ser realizado após o término do envio de todos os blocos da Curva, através do comando Recalcular Checksum de Curva (0x42).

- Se o comando foi enviado do Mestre para o Nó:
  - Caso os dados do comando sejam válidos, **deve-se** retornar o comando OK (0xE0);
  - Caso o ID da Curva seja inválido, **deve-se** retornar o comando ID Inválido (0xE3);
  - Caso o *offset* do bloco seja inválido (maior que o número de blocos da Curva), **deve-se** retornar o comando Valor Inválido (0xE4);

O exemplo ilustra a transmissão do quinto bloco da Curva de ID 3 cujos *bytes* têm todos o valor 0xDD.

Carga				
ID da Curva	*Offset* do bloco	Primeiro *byte* do bloco	...	Último *byte* do bloco

Tabela 38- Estrutura do comando Bloco de Curva

Cabeçalho		**Carga**				
41	FF	03	04	DD	...	DD

Tabela 39- Exemplo de comando Bloco de Curva

4.5.3 Recalcular Checksum de Curva (0x42)

COMANDO: 0x08
OBRIGATÓRIO?: Sim
DESCRIÇÃO: Pedido para que o Nó recalcule o Checksum da curva de escrita especificada.
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 1
ESTRUTURA DA CARGA: Tabela 40
EXEMPLO: Tabela 41

Este comando faz com que o Checksum da Curva especificada seja recalculado pelo Nó. Deve-se enviar este comando ao Nó toda vez que uma Curva de escrita for alterada.

- Caso os dados do comando sejam válidos, **deve-se** retornar o comando OK (0xE0);
- Caso o ID da Curva seja inválido, **deve-se** retornar o comando ID Inválido (0xE3);

O exemplo ilustra o pedido de recálculo do Checksum da curva de ID 0.

Carga
ID da Curva

Tabela 40- Estrutura do comando Recalcular Checksum de Curva

Cabeçalho		**Carga**
42	01	00

Tabela 41- Exemplo de comando Recalcular Checksum de Curva

4.6 Comandos para Controle da Comunicação Serial (0xD_)

Estes comandos devem ser implementados apenas por Nós que se utilizem da rede Serial para comunicação.

4.6.1 Consultar Lista de Grupos de *Multicast* (0xD0)

COMANDO: 0xD0
OBRIGATÓRIO?: Sim, se o Nó se comunicar via Serial.
DESCRIÇÃO: Pedido da lista de Grupos de *Multicast* aos quais o Nó pertence
SENTIDO: Mestre → Nó

A resposta a esse comando deve ser o comando Lista de Grupos de Multicast (0xD1).

4.6.2 Lista de Grupos de *Multicast* (0xD1)

COMANDO: 0xD1
OBRIGATÓRIO?: Sim, se o Nó se comunicar via Serial.
DESCRIÇÃO: Lista de Grupos de *Multicast* aos quais o nó pertence
SENTIDO: Nó → Mestre
TAMANHO DA CARGA: (número de Grupos de *Multicast* inscritos)
ESTRUTURA DA CARGA: Tabela 42
EXEMPLO: Tabela 43

O exemplo consiste na resposta dada por uma PUC pertencente apenas ao grupo padrão de *broadcast.*

Carga		
Endereço do primeiro Grupo de *Multicast*	…	Endereço do útimo Grupo de *Multicast*

Tabela 42- Estrutura do comando Lista de Grupos de Multicast

Cabeçalho		**Carga**
D1	03	FF

Tabela 43- Exemplo de comando Lista de Grupos de Multicast

4.6.3 Inscrever em Grupo de *Multicast* (0xD2)

COMANDO: 0xD2
OBRIGATÓRIO?: Não
DESCRIÇÃO: Pedido de inscrição em um Grupo de *Multicast*
SENTIDO: Mestre → Nó
TAMANHO DA CARGA: 1
ESTRUTURA DA CARGA: Tabela 44
EXEMPLO: Tabela 45

Este comando faz com que o destinatário se inscreva no Grupo de *Multicast* especificado.

- Caso a inscrição aconteça com sucesso, deve-se retornar o comando OK (0xE0);
- Caso o endereço passado esteja fora da faixa de endereços *multicast* permitidos ou caso o nó já esteja inscrito no Grupo desejado deve-se retornar o comando ID Inválido (0xE3);
- Caso o tamanho da carga seja diferente de 1, deve-se responder com o comando Tamanho da Carga Inválido (0xE5).

O exemplo faz a inscrição no Grupo de *Multicast* 240.

Carga
Endereço do Grupo

Tabela 44- Estrutura do comando Inscrever em Grupo de Multicast

Cabeçalho		**Carga**
D2	01	F0

Tabela 45- Exemplo de comando Inscrever em Grupo de Multicast

4.6.4 Desinscrever de Todos os Grupos de *Multicast* (0xD4)

COMANDO: 0xD4
OBRIGATÓRIO?: Não
DESCRIÇÃO: Pedido de desinscrição em todos os Grupos de *Multicast*
SENTIDO: Mestre → Nó

Este comando faz com que o destinatário se desinscreva de todos os Grupos de *Multicast* dos quais ele fazia parte. Deve-se retornar o comando OK (0xE0).

4.6.5 Ping (0xD6)

COMANDO: 0xD6
OBRIGATÓRIO?: Sim, se o Nó se comunicar via Serial
DESCRIÇÃO: Comando para teste de conectividade e latência
SENTIDO: Mestre → Nó ou Nó → Mestre
TAMANHO DA CARGA: 8 + (Tamanho da carga de teste)
ESTRUTURA DA CARGA: Tabela 46
EXEMPLO: Tabela 47

O comando de Ping é utilizado para se testar a conectividade e a latência entre Nós Seriais. O Meste envia um comando de Ping contendo sua medida de tempo atual e espera até que o Nó devolva o comando enviado, podendo, assim, medir o tempo de viagem (*Round-Trip Time*, *RTT*). O valor de Tempo deve ser de 8 *bytes*. A Carga de Teste pode variar entre 0 (zero) e 16378 *bytes*.

- Caso o comando esteja bem formado, **deve-se** responder com um comando Ping (0xD6) **exatamente** igual ao recebido, com o mesmo valor de Tempo e de Carga de Teste.
- Caso o tamanho da Carga seja menor que 8 *bytes*, deve-se retornar o comando Tamanho da Carga Inválido (0xE5).

No exemplo apresenta-se um comando Ping com sete *bytes* 0xAA como Carga de Teste e com tempo do Mestre marcando 0 (zero).

Carga										
Tempo (mais sig.)							Tempo (menos sig.)	Carga de Teste (primeiro *byte)*	...	Carga de Teste (último *byte*)

Tabela 46- Estrutura do comando Ping

Cabeçalho		**Carga**													
D6	0F	00	00	00	00	00	00	00	00	AA	AA	AA	AA	AA	AA	AA

Tabela 47- Exemplo de comando Ping

4.7 Comandos de Erro (0xE_)

Todos os Comandos de Erro obedecem o sentido Nó → Mestre e não possuem carga útil.

4.7.1 OK (0xE0)

Indica que o último comando enviado foi executado com sucesso.

4.7.2 Mensagem Mal Formada (0xE1)

Indica que o formato da última mensagem recebida difere do formato esperado.

4.7.3 Operação Não Suportada (0xE2)

Indica que o comando requisitado não foi implementado.

4.7.4 ID Inválido (0xE3)

Indica que foi passado um ID inválido no comando anterior.

4.7.5 Valor Inválido (0xE4)

Indica que um valor passado no comando anterior não está dentro da faixa de valores aceitáveis.

4.7.6 Tamanho da Carga Inválido (0xE5)

A Carga da última mensagem recebida tinha um tamanho diferente do esperado.

4.7.7 Somente Leitura (0xE6)

Foi tentada uma escrita em uma entidade de leitura.

4.7.8 Memória Insuficiente (0xE7)

O comando anterior falhou por falta de memória disponível.

4.7.9 Erro Interno (0xE8)

Houve um erro interno no *firmware.* O código deve ser checado por falhas de lógica.

5 Apêndice

5.1 Racionalização do campo TAMANHO do Protocolo

Cada módulo de uma PUC pode ter, no máximo, 8 conversores analógicos ou 6 portas digitais. Cada conversor analógico precisa de 3 *bytes* para ter seu valor armazenado. Cada porta digital precisa de 1 *byte* para seu valor. Cada PUC pode conter até 4 desses módulos.

A PUC com a maior necessidade de armazenamento de valores seria uma PUC com 4 módulos de 8 conversores analógicos cada, totalizando 96 *bytes* necessários para seus valores. Neste caso, a maior Mensagem do Protocolo possível de ser formada (excluindo-se os Comandos de Transferência de Curvas (0x4_)) seria com o comando Escrever em Grupo de Variáveis (0x22) que ajustasse todos os conversores analógicos. A Carga Útil de tal mensagem seria de 1 *byte* para identificar o grupo de conversores e 96 *bytes* contendo os ajustes, totalizando 97 *bytes* de carga. O valor 97 é facilmente representado por uma variável de 1 *byte*, que pode representar valores de 0 a 255. Assim, o campo TAMANHO poderia ter apenas 1 *byte* e seria capaz de representar todas as Mensagens do Protocolo aplicado às PUC's.

Porém, existe a necessidade de se transferir Curvas usando o Protocolo (curvas de Post-Mortem, Ciclagem, Rampa, etc). Com essa necessidade em mente, consideram-se algumas possibilidades para o campo TAMANHO das Mensagens do Protocolo e as vantagens e desvantagens de cada uma.

Nos cálculos a seguir:

- Considera-se como exemplo a transferência de uma curva de 65536 *bytes*, de um Nó para o Mestre.
- Considera-se que o cabeçalho TCP/IP é de 40 *bytes* (20 *bytes* TCP, 20 *bytes* IP) e que o MTU é 1500 *bytes*, resultando na transferência máxima de 1460 *bytes* por pacote.
- Considera-se que o cabeçalho na transmissão Serial é de 5 *bytes* (como descrito na seção Comunicação Serial.

1. TAMANHO de 1 *byte*

O campo TAMANHO ter 1 *byte* implica em Cargas Úteis de Mensagem de até 255 *bytes*. Neste caso, para transferir 65536 *bytes* da curva, são necessárias 258 Mensagens. Porém, dada a natureza do Protocolo, para cada uma das 258 Mensagens enviadas pelo Nó devem existir 258 pedidos de Mensagem feitos pelo Mestre, totalizando 516 Mensagens. Calculam-se os *overheads*:

- *Overhead* de Protocolo: 2 *bytes* por Mensagem: 1032 *bytes;*
- *Overhead* da Rede Serial: 5 *bytes* por Mensagem: 2580 *bytes;*

  Totalizando 1032 + 2580 = 3612 *bytes*, representa **5.51%** da curva sendo transmitida.
- *Overhead* da Rede TCP/IP: 40 *bytes* por Mensagem: 20640 *bytes;*

  Totalizando 1032 + 20640 = 21672 *bytes*, representa **33.07%** da curva sendo transmitida.

2. **TAMANHO de 2 *bytes***

Um campo de TAMANHO de 2 *bytes* permite que as Mensagens do Protocolo tenham até 65535 *bytes* de Carga Útil. Neste caso, seriam necessárias 2 Mensagens para transferir a curva hipotética, mais 2 Mensagens de pedido, totalizando 4 Mensagens. Porém, no caso TCP/IP, deve-se lembrar que sua carga útil máxima é de 1460 *bytes*, dos quais 3 devem ser reservados para o cabeçalho do Protocolo (1 *byte* de COMANDO, 2 *bytes* de TAMANHO), reduzindo a carga útil para 1457 *bytes* por pacote. Assim, são necessárias 45 Mensagens de transmissão, que somadas às 45 Mensagens de pedido totalizam 90 Mensagens.

- *Overhead* de Protocolo (Serial): 3 *bytes* por Mensagem, 4 Mensagens: 12 *bytes;*
- *Overhead* de Protocolo (TCP/IP): 3 *bytes* por Mensagem, 90 Mensagens: 270 *bytes;*
- *Overhead* da Rede Serial: 5 *bytes* por Mensagem, 4 Mensagens: 20 *bytes;*

  Totalizando 12 + 20 = 32 *bytes*, representa **0.05%** da curva sendo transmitida.
- *Overhead* da Rede TCP/IP: 40 *bytes* por Mensagem, 90 Mensagens: 3600 *bytes;*

  Totalizando 270 + 3600 = 3870 *bytes*, representa **5.90%** da curva sendo transmitida.

3. **TAMANHO de 1 *byte,* com um campo de 1 *bit* e um campo de 7 *bits***

Nesta solução trata-se o TAMANHO como a união de dois campos de *bits*:

- O *bit* mais significativo, chamado aqui de *m* (de “modo”), indica o modo como os *bits* seguintes devem ser interpretados.
- Os *bits* restantes, chamados *n* (de “número”) contém o tamanho da carga útil, em duas diferentes codificações.

O valor do TAMANHO é, então, tido segundo a Equação 2.

$$\begin{cases} \text{Se } m = 0, \text{TAMANHO} = n \\ \text{Se } m = 1, \text{TAMANHO} = 128 * (n + 1) \end{cases}$$

Equação 2- Possível interpretação do campo TAMANHO

Assim, são necessárias apenas 4 Mensagens para transmitir a curva hipotética inteira, que somadas às 4 Mensagens de pedido, totalizam 8 Mensagens para a transferência em Serial. Em TCP/IP, contudo, são necessárias ainda 45 Mensagens, totalizando 90.

- *Overhead* de Protocolo (Serial): 2 *bytes* por Mensagem, 8 Mensagens: 16 *bytes;*
- *Overhead* de Protocolo (TCP/IP): 2 *bytes* por Mensagem, 90 Mensagens: 180 *bytes;*

- *Overhead* da Rede Serial: 5 *bytes* por Mensagem, 8 Mensagens: 40 *bytes;*

  Totalizando 16 + 40 = 56 *bytes*, representa **0.08%** da curva sendo transmitida.

- *Overhead* da Rede TCP/IP: 40 *bytes* por Mensagem, 90 Mensagens: 3600 *bytes;*

  Totalizando 180 + 3600 = 3780 *bytes*, representa **5.77%** da curva sendo transmitida.

Um comparativo do *overhead* das soluções com diferentes quantidades de *bytes* a serem transferidos (considerando pedido e resposta, como nos cálculos anteriores) é apresentado na Tabela 48.

Porcentagem de *overhead*												
Solução		**Quantidade de *bytes* a serem transmitidos**										
		16	**64**	**128**	**256**	**512**	**1k**	**4k**	**16k**	**32k**	**64k**	**1M**	**4M**
Serial	**1**	87.50	21.87	10.93	10.94	8.20	6.83	5.81	5.55	5.51	5.51	5.49	5.49
	2	100.00	25.00	12.5	6.25	3.12	1.56	0.39	0.10	0.05	0.05	0.03	0.02
	3	87.50	21.87	10.94	5.47	2.73	1.37	0.34	0.08	0.08	0.08	0.08	0.08
TCP/IP	**1**	525.00	131.25	65.62	65.62	49.21	41.01	34.86	33.32	33.07	33.07	32.95	32.94
	2	537.50	134.37	67.19	33.59	16.80	8.40	6.30	6.30	6.03	5.90	5.90	5.90
	3	525.00	131.25	65.62	32.81	16.41	8.20	6.15	6.15	5.90	5.77	5.77	5.77

Tabela 48- Comparativo do overhead na transmissão de diferentes quantidades de bytes

Assim, analisando os dados da tabela, pode-se descartar a solução 1 por apresentar um *overhead* muito maior do que as outras soluções para pacotes grandes. A solução 2 apresenta um *overhead* menor que a solução 3 para pacotes grandes, mas perde para a terceira solução nos pacotes menores.

Levando-se em conta que a maioria das transmissões (nas redes de PUC's) serão de pacotes pequenos, foi escolhida a terceira solução para o TAMANHO das Mensagens do Protocolo.