HI tecnologia
Indústria e Comércio Ltda

Notas de Software

DLL SS2SCA - Funções básicas de acesso ao SCP Server II

Compatível com DLL
Versão 1.4.XX

HI Tecnologia

Documento de acesso público

PNS.00030 **Versão 1.02** **março-2014**

Apresentação

Este documento apresenta API de funções de interface com a DLL básica de acesso ao servidor de comunicação SCP Server II.

Esta nota de software foi elaborada pela **HI Tecnologia Indústria e Comércio Ltda**. Quaisquer dúvidas ou esclarecimentos sobre as informações contidas neste documento podem ser obtidas diretamente com o nosso departamento de suporte a clientes, através do telefone (19) 2139-1700 ou do e-mail "suporte@hitecnologia.com.br". Favor mencionar as informações a seguir para que possamos identificar os dados relativos a este documento.

ID da Nota de Software:	PNS.00030
Versão Documento:	1.02

HI Tecnologia Indústria e Comércio Ltda.

Endereço: Av. Dr. Armando de Sales Oliveira, 445. Bairro Taquaral.

Cidade: Campinas – SP
CEP: 13076-015

Fone: +55 (19) 2139-1700
Fax: +55 (19) 2139-1710

Web site: www.hitecnologia.com.br

Perguntas Freguentes *FAQ*: faq.webhi.com.br

E-mail: *Vendas*: vendas@hitecnologia.com.br
Suporte técnico: suporte@hitecnologia.com.br
Engenharia de aplicação: engenharia@hitecnologia.com.br

HI tecnologia Indústria e Comércio Ltda

DLL SS2SCA - Funções básicas de acesso ao SCP Server II

Referência: PNS.00030 **Revisão:** 2
Arquivo: PNS0003000.doc **Atualizado em:** 30/10/2012

Índice

1. Introdução 4
1.1 Informação *Copyright* 5
1.2 Disclaimer 5
1.3 Novas versões 5
1.4 Sugestões 5
2. Identificação da DLL de comunicação 5
3. Distribuição da DLL 5
4. Endereçamento do controlador via DLL SS2SCA 6
4.1 Endereço do controlador 6
4.1.1 Endereço Global 6
4.2 Acesso ao controlador em uma conexão ponto a ponto 6
4.3 Acesso ao controlador em uma conexão em rede 6
4.4 Drivers e Canais no SCP Server II 7
4.5 Remapeamento de endereços 8
4.5.1 Remapeamento para os drivers serial e UDP Broadcast 8
4.5.2 Remapeamento para o driver ethernet 9
4.6 Definição do endereço do controlador a ser utilizado na DLL 9
5. Lista de funções da DLL 11
6. Descrição das funções 11
6.1 SCPRelease 12
6.2 SCPResetDriver 13
6.3 SCPOpenPort 14
6.4 SCPClosePort 15
6.5 SCPForceConnection 16
6.6 SCPCheckConnection 17
6.7 SCPReadData 18
6.8 SCPWriteData 20
7. Definições do protocolo 22
8. Códigos de retorno 22
8.1 Referências 22
Controle do Documento 23
Considerações gerais 23
Responsabilidades pelo documento 23

1. Introdução

Os controladores da HI Tecnologia possuem recursos para comunicação através de interfaces seriais RS232-C, RS485 e Ethernet. Estes canais de comunicação são utilizados para programação, depuração e supervisão das informações contidas no programa do equipamento.

Utilizando um protocolo de aplicação desenvolvido pela HI Tecnologia, denominado SCP-HI, é possível, utilizando qualquer dos recursos de comunicação, acessar toda a base de dados da aplicação nos PLC's, permitindo programação e supervisão de qualquer dos parâmetros do programa.

Para facilitar o acesso aos controladores e possibilitar compartilhar este acesso com múltiplos usuários simultaneamente, foi desenvolvido o servidor de comunicação SCP Server II. Com este pacote é possível conectar-se simultaneamente a múltiplos equipamentos com qualquer um dos recursos de comunicação (RS232-C, RS485 ou Ethernet).

A figura 1 apresenta um cenário genérico de utilização do SCP Server II dentro de uma rede local.

Os usuários 1 a N podem utilizar uma mesma máquina, ou máquinas distintas em uma rede local. O SCP Server II pode estar instalado em qualquer das máquinas da rede, ou seja, uma das máquinas utilizadas pelos usuários ou em uma máquina separada.

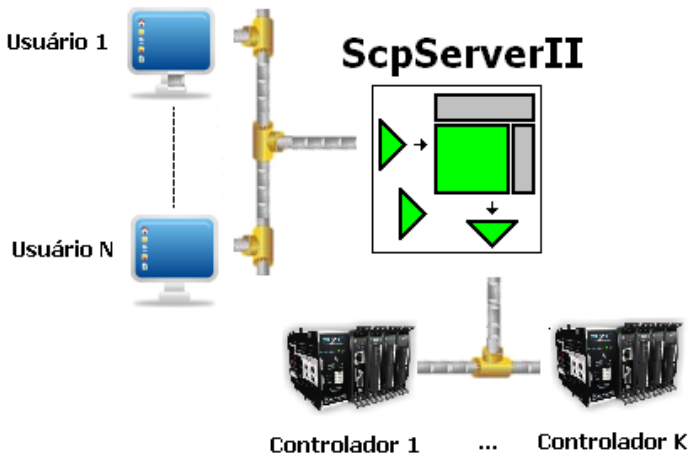


Figura 1

Os *drivers* de acesso aos controladores são criados e configurados no SCP Server II que é responsável por toda gerência de acesso aos equipamentos com qualquer dos recursos de comunicação disponíveis.

Uma vez corretamente configurado, o SCP Server II aguarda a conexão (ethernet) de clientes que desejam acessar os equipamentos através dos *drivers* criados. Desta forma, vários clientes podem obter acesso às bases de dados dos controladores simultaneamente, e de forma transparente ao protocolo e ao recurso de comunicação utilizado.

Para possibilitar acesso a todos estes recursos, a aplicação do usuário necessita conectar-se, especificar os *drivers* a serem utilizados e enviar os comandos necessários para a troca de dados com os controladores. Este documento apresenta a API de funções disponível na DLL SS2SCA. DLL, desenvolvida para ser utilizada pela aplicação do usuário de forma a permitir, de uma maneira simples, a conexão e utilização dos recursos disponibilizados pelo SCP Server II para acesso à base de dados dos controladores.

Esta DLL disponibiliza os recursos mínimos necessários para acesso aos controladores. O desenvolvimento desta DLL visa facilitar a migração de aplicações de clientes que já utilizam as DLLs de comunicação SCPHIV9 e SCPHIV10 para utilizarem os recursos do SCP Server II com o mínimo de esforço possível.

Com esta abordagem, existem inúmeras funcionalidades no SCP Server II que não estão acessíveis por esta DLL, estando disponíveis em outras DLLs da HI Tecnologia.

Neste documento são apresentadas as funções de interface desta DLL, definindo funcionalidades, parâmetros, e sintaxe para sua correta utilização. Pode ser obtido, junto à HI Tecnologia, um programa desenvolvido em C++ (Builder) que exemplifica como utilizar esta DLL.

1.1 Informação *Copyright*

1.2 Disclaimer

A utilização dos conceitos, exemplos e outros elementos deste documento é responsabilidade exclusiva do usuário.

1.3 Novas versões

Novas versões desta DLL podem ser liberadas. Para verificar a disponibilidade de novas versões, consulte a HI Tecnologia através do e-mail suporte@hitecnologia.com.br.

1.4 Sugestões

Sugestões são sempre bem vindas, e a HI Tecnologia agradece aqueles que nos auxiliam no aprimoramento das informações contidas neste documento. Por favor, envie seus comentários e sugestões para suporte@hitecnologia.com.br.

2. Identificação da DLL de comunicação

Este documento descreve as funções básicas de acesso aos controladores disponíveis na DLL SS2SCA.DLL. Esta DLL provê uma função para obtenção da versão e revisão corrente. Até a data de atualização deste documento, a versão corrente da DLL é 1.4.00.

3. Distribuição da DLL

A DLL SS2SCA é distribuída com os seguintes elementos:

Arquivo	Descrição
SS2SCA.dll	DLL de comunicação SS2SCA versão 1.4.xx
SS2SCA.lib	Biblioteca de interface. Utilizado para link estático com aplicações desenvolvidas em C++ Builder XE.
Hi_Defines.h	Arquivo de include com definições de constantes utilizadas.
PNS.00030	Nota de software documentando as funções básicas da DLL de acesso aos controladores via o ScpServerII (este documento)

4. Endereçamento do controlador via DLL SS2SCA

Para possibilitar o acesso a vários controladores simultaneamente, sem alterar a interface da DLL, o parâmetro que especifica o endereço do controlador passou a incorporar informações adicionais. Este novo formato para endereçamento do controlador é definido a seguir.

4.1 Endereço do controlador

Todos os controladores da HI possuem o mesmo modelo de endereçamento de comunicação. Existe um parâmetro no controlador (que pode ser alterado), especificando um número entre 1 e 255 (ou 1 a 1023, quando os equipamentos possuem firmware G3), que é utilizado para identificar o equipamento quando conectado a uma rede de comunicação qualquer.

Desta forma, todo pacote de comunicação que chega a um equipamento possui um campo que especifica o endereço do controlador ao qual o pacote pertence. Se o endereço do pacote for igual ao endereço do controlador, este pacote é processado pelo equipamento, caso contrário, ele é descartado. Todos os controladores da HI Tecnologia são configurados em fábrica com o endereço de comunicação número 1. Através do ambiente de programação dos controladores, este endereço pode ser modificado pelo usuário.

4.1.1 Endereço Global

Existe um endereço do controlador que, quando utilizado, é tratado de forma diferenciada. O endereço 255, quando recebido em um pacote de comunicação por um equipamento, indica para o mesmo que o pacote deve ser validado e, conseqüentemente, tratado independentemente do endereço definido no controlador. Desta forma, um pacote com endereço 255 é sempre tratado pelo equipamento que o recebeu. Por este motivo, este endereço não deve nunca ser utilizado em um cenário de comunicação em rede.

4.2 Acesso ao controlador em uma conexão ponto a ponto

Quando o programa de aplicação estiver conectado a um único controlador, qualquer endereço válido (1... 255, ou 1 a 1023 quando os equipamentos possuem firmware G3) pode ser configurado no equipamento. Neste caso, o programa de aplicação poderá montar os pacotes de comunicação com o endereço configurado no controlador ou com o endereço 255.

Figura 2

4.3 Acesso ao controlador em uma conexão em rede

Consideremos agora um cenário onde um programa de aplicação está conectado a 3 controladores via uma rede RS485, conforme indicado na figura 3. Neste contexto, para uma operação correta do sistema de comunicação, as seguintes restrições devem ser respeitadas:

- Dois ou mais equipamentos participantes da rede não podem possuir o mesmo endereço.
- Nenhum equipamento deve ser configurado com o endereço 255.
- O programa de aplicação nunca deve enviar pacotes na rede com o endereço 255.

OBS: Normalmente, quando utilizada uma arquitetura de comunicação em rede, deve-se evitar se possível a utilização do endereço 1 para os equipamentos. Este procedimento facilita o processo de adição de novos equipamentos à rede, pois, como já mencionado, os controladores são fornecidos de fábrica com o endereço 1.

Outra vantagem reside no fato de que, caso ocorra uma falha no controlador e o mesmo seja reinicializado, este passará a estar configurado com o endereço 1, e caso já exista um equipamento na rede com este endereço, ocorrerá conflito no tratamento dos pacotes, gerando falha no acesso aos dois controladores.

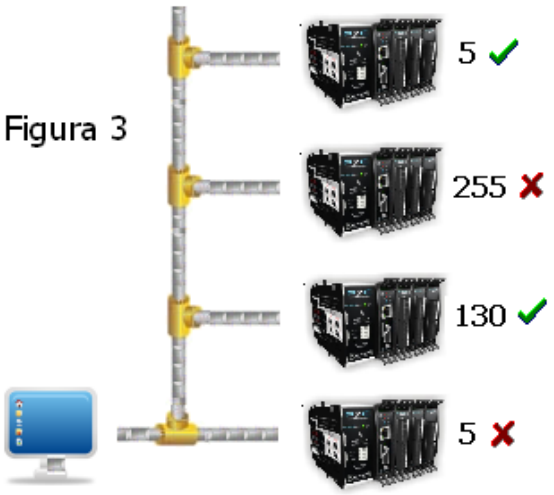


Figura 3

4.4 Drivers e Canais no SCP Server II

O SCP Server II suporta operação simultânea com múltiplos recursos de comunicação. Sendo assim, é possível, por exemplo, configurar o servidor para acessar um controlador utilizando a porta serial COM1 e acessar um segundo controlador através de uma conexão ethernet TCP/IP, ou ainda acessar um conjunto de controladores via uma conexão ethernet UDP/Broadcast.

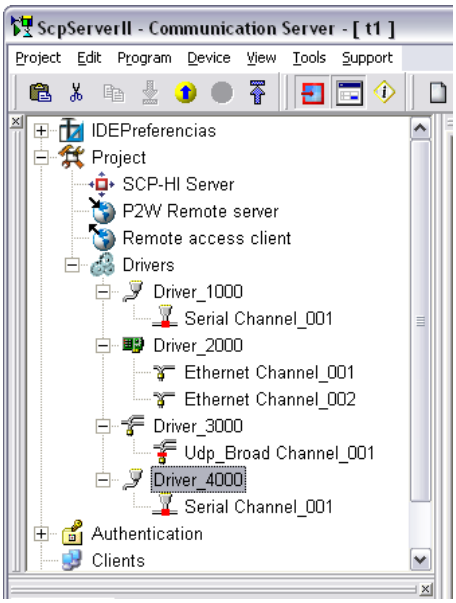
Para cada recurso de comunicação que se deseja utilizar no SCP Server II, deve-se criar um ou mais *drivers* de comunicação. Cada *driver* criado, dependendo do seu tipo, pode suportar 1 ou mais canais de comunicação.

Existem 3 tipos de *drivers* disponíveis no SCP Server II, conforme apresentado na tabela a seguir:

Driver	Canais (Max)	Descrição
Serial	1	Disponibiliza acesso a um ou mais controladores através de uma porta serial (COM). Esta porta pode estar conectada ponto a ponto a um único controlador, ou a uma rede de controladores via um canal RS485 ou um sistema de rádio de dados.
Ethernet TCP/IP - UDP	50	Disponibiliza acesso a até 50 equipamentos em uma rede ethernet. Cada conexão com um equipamento utiliza um canal independente e pode ser configurada com protocolo de transporte TCP/IP ou UDP.
Ethernet UDP Broadcast	1	Disponibiliza acesso a um ou mais controladores através de uma conexão ethernet, utilizando como protocolo de transporte UDP Broadcast. Permite configurar acesso a uma rede de controladores através de um único canal ethernet. Alguns sistemas de rádio de dados com interface ethernet suportam apenas este modo de operação.

A figura ao lado apresenta a interface de configuração do SCP Server II com 4 *drivers* criados. Sempre que um *driver* é criado, lhe é atribuído um identificador único múltiplo de 1000. Ou seja:

- O primeiro *driver* criado recebe o identificador 1000;
- O segundo o identificador 2000;
- O terceiro 3000, e assim, sucessivamente.



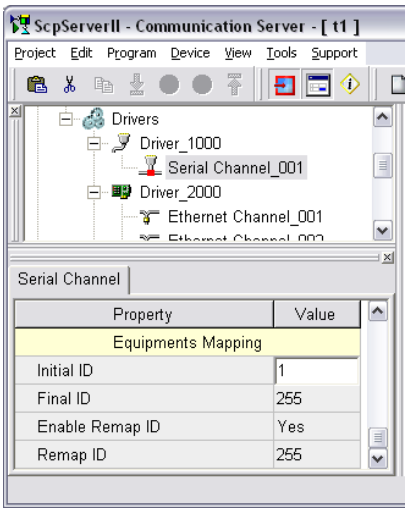
Com este número podemos identificar um dado *driver* dentro do servidor de comunicação. Esta informação é importante e será utilizada pela DLL para identificar para qual dos *drivers* presentes no servidor, um determinado pacote deverá ser transferido.

Múltiplos *drivers* do mesmo tipo podem ser criados. A versão corrente do SCP Server II, na data desta publicação suportava um máximo de 32 *drivers* por servidor.

4.5 Remapeamento de endereços

O SCP Server II disponibiliza recursos para "remapear" o endereço do controlador recebido pelo programa de aplicação. Desta forma, o programa de aplicação pode definir um endereço fora da faixa utilizada pelo controlador. De acordo com a configuração do canal associado, este endereço é convertido para um novo valor dentro da faixa válida (1 a 255) (ou 1 a 1023 quando os equipamentos possuem firmware G3) para ser enviado no pacote que é transmitido para o equipamento.

Na base de configuração de cada um dos canais associados aos *drivers,* existe um conjunto de parâmetros para realizar o remapeamento, conforme indicado na figura ao lado.



4.5.1 Remapeamento para os drivers serial e UDP Broadcast

Para não utilizar o remapeamento, no caso do *driver* serial, utilize a seguinte configuração para os parâmetros:

Parâmetro	Valor	Obs.
Initial ID	1	Valida toda a faixa de endereços dos controladores.
Final ID	255	Obs.: 1023 quando os equipamentos possuírem firmware G3.
Enable Remap ID	No	Remapeamento desabilitado.
Remap ID	x	

Com a configuração acima, o endereço enviado pelo programa de aplicação é repassado sem nenhuma alteração para o(s) equipamento(s) associado(s) ao canal.

Por exemplo, para remapear o endereço do programa de aplicação 1001 para o endereço do controlador 1, configure os parâmetros, conforme a tabela a seguir.

Parâmetro	Valor	Obs.
Initial ID	1001	Valida toda a faixa de endereços dos controladores
Final ID	1250	
Enable Remap ID	Yes	Remapeamento habilitado
Remap ID	1	

Note que na configuração acima, o endereço 1001 será remapeado para 1, o endereço 1002 será remapeado para 2, e assim sucessivamente. Não é possível a utilização dos endereços 1251->251 a 1255 -> 255, pois foi definido em endereço final igual a 1250.

4.5.2 Remapeamento para o driver ethernet

No caso do *driver* ethernet, cada canal de comunicação está associado a um único equipamento através do seu endereço IP. Desta forma, os parâmetros *Initial ID* e *Final ID* devem ser iguais e devem especificar o endereço recebido pelo programa de aplicação, conforme exemplificado na tabela a seguir:

Parâmetro	Valor	Obs.
Initial ID	5	Associa o endereço 5 ao equipamento do canal
Final ID	5	
Enable Remap ID	Yes	Remapeamento habilitado
Remap ID	255	

Note que na configuração acima, o remapeamento foi habilitado e definido como endereço 255. Esta abordagem pode ser realizada (e é aconselhável), uma vez que existe apenas um endereço associado ao canal ethernet.

4.6 Definição do endereço do controlador a ser utilizado na DLL

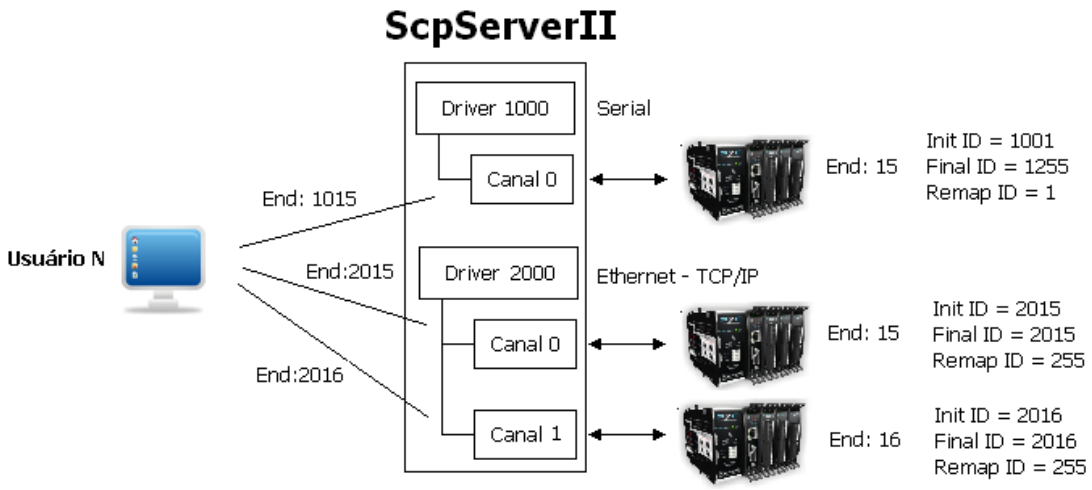
Para identificar qual endereço do controlador deve ser utilizado nas chamadas das funções da DLL, considere o seguinte exemplo:

- Um programa de aplicação que utiliza a DLL deseja comunicar-se com 3 equipamentos simultaneamente.
- O primeiro equipamento está conectado ao servidor de comunicação, através do canal serial, e possui o endereço de comunicação igual a 15.
- O segundo equipamento está conectado ao servidor de comunicação através de um canal ethernet com protocolo de transporte TCP/IP, e também possui o endereço de comunicação igual a 15.
- O terceiro equipamento está conectado ao servidor de comunicação através de um canal ethernet com protocolo de transporte TCP/IP, e possui o endereço de comunicação igual a 16.

Como existem recursos de comunicação diferentes, o SCP Server II irá necessitar de no mínimo 2 *drivers* distintos para conectar-se a todos os equipamentos. Neste caso, uma opção seria criar um *driver* 1000 do tipo serial e um *driver* 2000 do tipo Ethernet. O *driver* serial deve possuir apenas um canal a ser configurado para

utilizar a porta COM conectada ao controlador 1 (controlador serial). O *driver* ethernet deve possuir, no mínimo, 2 canais para que possam ser configurados os endereços IP e porta dos controladores 2 e 3.

Sendo assim, uma configuração possível do servidor seria a apresentada na figura a seguir:



As chamadas de comunicação da DLL que necessitam do parâmetro “endereço do controlador” irão utilizar os endereços 1015 para acessar o controlador 1, 2015 para acessar o controlador 2 e 2016 para acessar o controlador 3. Os parâmetros de remapeamento de cada um dos canais estão indicados na figura.

Note que:

- Para os equipamentos ethernet, os endereços inicial e final são os mesmos, indicando que apenas um equipamento está associado ao canal correspondente. Adicionalmente, o endereço de remapeamento foi definido como 255. Desta forma, é irrelevante o endereço definido no controlador associado ao canal ethernet, ou seja, a comunicação será bem sucedida com qualquer endereço que o controlador estiver configurado.

- Para o canal serial, a seguinte configuração também seria válida (Init ID = 1015, Final ID = 1015, Remap ID=15). Entretanto, o remapeamento define toda a faixa de endereços e não apenas o endereço do controlador associado (15). Com esta abordagem, caso o endereço do controlador seja modificado ou outros controladores sejam adicionais ao *driver*, não existirá necessidade de alteração da configuração do servidor SCP Server II.

5. Lista de funções da DLL

A tabela a seguir apresenta a lista das funções disponíveis na DLL para utilização pelo usuário.

Nro	Nome	Categoria	Versão	Obs.
01	*SCPRelease*	Controle / Informação do *driver*	1.2.00	Funcional
02	*SCPResetDriver*	Controle / Informação do *driver*	1.2.00	Funcional
03	*SCPOpenPort*	Controle / Informação do *driver*	1.2.00	Funcional
04	*SCPClosePort*	Controle / Informação do *driver*	1.2.00	Funcional
05	*SCPForceConnection*	Gerência de Comunicação	1.2.00	Funcional
06	*SCPCheckConnection*	Gerência de Comunicação	1.2.00	Funcional
07	*SCPReadData*	Comunicação (Troca de dados)	1.2.00	Funcional
08	*SCPWriteData*	Comunicação (Troca de dados)	1.2.00	Funcional

6. Descrição das funções

Este capítulo identifica e define a interface para utilização das funções disponíveis na DLL SS2SCA.DLL Cada função é documentada com os seguintes itens:

- Identificação do Nome;
- Protótipo da função em C/C++;
- Descrição da funcionalidade associada;
- Especificação e descrição dos parâmetros de entrada (quando aplicável);
- Especificação e descrição dos parâmetros de saída (quando aplicável);
- Descrição do retorno da função (quando aplicável);
- Notas adicionais, quando aplicável.

6.1 SCPRelease

Função 01 **SCPRelease**

Protótipo da função em C/C++

```
int WINAPI SCPRelease ( int &Versao, int &Revisao )
```

Descrição

Obtêm da DLL a versão e revisão corrente do mesmo. A revisão é dividida em 2 partes (revisão maior e revisão menor). O formato geral da versão utilizado é:
V.J.MM, onde:
V = Nro da versão;
J = Nro. da revisão maior (1 dígito);
M = Nro. da revisão menor (2 dígitos).

Exemplo: Quando retornados os valores Versão = 1 e Revisão = 203, a versão do pacote será:
1.2.03

Parâmetros de Entrada

Versão	Número da versão do *driver*
Revisão	Número da revisão do *driver*

Parâmetros de Saída

Não se aplica

Código de Retorno

= 0 (SUCCESS)	Indica função executada com sucesso.
<> 0	Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090.

Notas

6.2 SCPResetDriver

Função 02 **SCPResetDriver**

Protótipo da função em C/C++

```
int WINAPI SCPResetDriver ( void )
```

Descrição

Fecha a conexão com o servidor e abre novamente, reinicializando a conexão com os parâmetros originais.

Parâmetros de Entrada

Não se aplica

Parâmetros de Saída

Não se aplica

Código de Retorno

= 0 (SUCCESS)	Indica função executada com sucesso.
<> 0	Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090.

Notas

6.3 SCPOpenPort

Função 03 **SCPOpenPort**

Protótipo da função em C/C++

```
int WINAPI SCPOpenPort (char* sServerIP, int nServerPort, int dummy)
```

Descrição

Cria uma conexão com o servidor de comunicação especificado pelo endereço IP e porta. Identifica e ativa todos os *drivers* existentes no servidor. Após a execução desta função com sucesso, a conexão está apta para trocar *frames* com todos os equipamentos acessados pelo servidor de comunicação SCP ServerII. Terminado o processo de comunicação, utilize a função ScpClosePort para fechar a conexão com o servidor de comunicação.

Parâmetros de Entrada

sServerIP	Endereço IP da máquina onde está o servidor de comunicação
nServerPort	Porta ethernet de acesso ao servidor de comunicação
nDummy	Parâmetro não utilizado. Mantido para compatibilidade com a versão anterior.

Parâmetros de Saída

Não se aplica

Código de Retorno

= 0 (SUCCESS)	Indica função executada com sucesso.
<> 0	Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090.

Notas

6.4 SCPClosePort

Função 04 **SCPClosePort**

Protótipo da função em C/C++

```
int WINAPI SCPClosePort ( void )
```

Descrição

Fecha a conexão entre a aplicação que utiliza a DLL e o servidor de comunicação. Esta ação fecha também todos os *drivers* abertos no processo de abertura SCPOpenPort.

Parâmetros de Entrada

Não se aplica

Parâmetros de Saída

Não se aplica

Código de Retorno

= 0 (SUCCESS)	Indica função executada com sucesso.
<> 0	Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090.

Notas Quando a DLL é liberada, esta fecha automaticamente a conexão com o servidor de comunicação, caso esta esteja aberta.

6.5 SCPForceConnection

Função 05 **SCPForceConnection**

Protótipo da função em C/C++

int WINAPI SCPForceConnetion (int client_device_id)

Descrição

Sempre que um controlador é acessado pela primeira vez, o SCP Server II, se configurado para tal, mantém um controle de conexão com o equipamento, de forma que sempre que houver uma falha de acesso ao controlador, o próprio SCP Server II periodicamente envia *frames* de conexão para o equipamento, com o objetivo de identificar quando o equipamento voltar a responder.

Durante este período de detecção de conexão, todos os frames de comunicação enviados retornaram com o código de falha _RCCS_DEVICE_OFFLINE (código 20385). Para forçar uma tentativa de reconexão do SCP Server II com o equipamento remoto, é utilizada esta função (SCPForceConnection). Quando enviada para o servidor, esta ativa imediatamente um comando de conexão com o equipamento remoto e, caso o resultado seja positivo, o servidor se torna apto a encaminhar novos frames de comunicação para o equipamento remoto.

Parâmetros de Entrada

Client_device_id	Identificador do equipamento pela aplicação. Este nro deve compor o identificador do *driver* do servidor associado ao equipamento a ser acessado e o endereço do equipamento. Por exemplo, para acessar o controlador com endereço 20, localizado no *driver* 3000, utilize para este parâmetro o valor 3020. Note que o canal do *driver* associado ao SCP Server II deve remapear este valor para o endereço corrente, ou seja, o endereço 3020 deve ser remapeado para 20. Se for utilizado um valor abaixo de 1000, a DLL considera que este endereço está mapeado no *driver* 1000, e encaminha o frame associado para este *driver*. (Vide capítulo 4 para esclarecimentos adicionais)

Parâmetros de Saída

Não se aplica

Código de Retorno

= 0 (SUCCESS)	Indica função executada com sucesso.
<> 0	Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090.

Notas

6.6 SCPCheckConnection

Função 06 **SCPCheckConnection**

Protótipo da função em C/C++

int WINAPI SCPCheckConnetion (int client_device_id)

Descrição

Envia um frame de teste de conexão para o controlador especificado. Esta função permite detectar a presença de controlador e avaliar se o *link* de comunicação está operacional. Como não existe troca efetiva de dados nesta função, existe um *timeout* específico para uma detecção mais rápida de uma conexão.

Parâmetros de Entrada

Client_device_id	Identificador do equipamento pela aplicação. Este nro deve compor o identificador do *driver* do servidor associado ao equipamento a ser acessado e o endereço do equipamento. Por exemplo, para acessar o controlador com endereço 20, localizado no *driver* 3000, utilize para este parâmetro o valor 3020. Note que o canal do *driver* associado no SCP Server II deve remapear este valor para o endereço corrente, ou seja, o endereço 3020 deve ser remapeado para 20. Se for utilizado um valor abaixo de 1000, a DLL considera que este endereço está mapeado no *driver* 1000 e encaminha o *frame* associado para este *driver*. (Vide capítulo 4 para esclarecimentos adicionais)

Parâmetros de Saída

Não se aplica

Código de Retorno

= 0 (SUCCESS)	Indica função executada com sucesso.
<> 0	Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090. .

Notas

6.7 SCPReadData

Função 07 **SCPReadData**

Protótipo da função em C/C++

```
int WINAPI SCPReadData (int client_device_id, int TypeVar, int VarIni, int VarQuant, void* DataBuf)
```

Descrição

Solicita ao controlador os valores das variáveis especificadas na função. O usuário deve especificar o tipo de variável (R, M, D ou L), identificar o número da variável inicial e a quantidade de variáveis a ser obtida a partir desta. Os valores das variáveis solicitadas são transferidos para o *buffer* de dados fornecido.

É de responsabilidade de o usuário prover o buffer de recepção com o espaço necessário para receber todos os dados solicitados. Cada tipo de dado requer um espaço distinto no buffer de recepção, conforme descrito na tabela a seguir:

Tipo da Variável	Tipo associado na linguagem C	Tamanho
R	*unsigned char*	1 byte (8 bits)
M	*short int*	2 bytes (16 bits)
D	*float*	4 bytes (32 bits)
L	*int*	4 bytes (32 bits)

O tamanho máximo de um *frame* de dados no protocolo SCP-HI é de 249 bytes. Isto define a quantidade máxima de variáveis de cada tipo que pode ser obtida do controlador em um único comando. Quando a quantidade de variáveis requisitadas ultrapassar este limite, a função divide o *buffer* de recepção, solicitando tantos *frames* de comunicação quanto forem necessários para obter todos os dados solicitados.

Parâmetros de Entrada

`Client_device_id` Identificador do equipamento pela aplicação. Este nro deve compor o identificador do *driver* do servidor associado ao equipamento a ser acessado e o endereço do equipamento.
Por exemplo, para acessar o controlador com endereço 20, localizado no *driver* 3000, utilize para este parâmetro o valor 3020. Note que o canal do *driver* associado ao SCP Server II deve remapear este valor para o endereço corrente, ou seja, o endereço 3020 deve ser remapeado para 20.
Se for utilizado um valor abaixo de 1000, a DLL considera que este endereço está mapeado no *driver* 1000 e encaminha o frame associado para este *driver*.
(Vide capítulo 4 para esclarecimentos adicionais)

`TypeVar` Identifica o tipo de variável a ser solicitada do controlador, conforme tabela a seguir:

TypeVar	Tipo de variável	ID	Tamanho	Faixa de valores	
				Min	Max
0	Contato auxiliar	R	1 byte	OFF(0)	ON(255)
1	Memórias inteiras	M	2 bytes	-32768	+32767
2	Memórias reais	D	4 bytes	-10E-38	+10E+38
3	Memórias long	L	4 bytes	-2147483648	+2147483647

`VarIni` Identifica o número da variável inicial a ser obtida. Todas as variáveis da base de dados dos controladores HI começam em zero (0) e vão até o número máximo de variáveis do tipo – 1. O número de variáveis de cada tipo depende do tipo de firmware carregado no controlador e, no caso de firmware de PLC, este número é alocado dinamicamente em função do programa de aplicação criado.
Caso seja especificada uma variável não definida no controlador, será retornado pela função um código de falha indicando esta situação.

`VarQuant` Especifica o número de variáveis a serem obtidas a partir de `VarIni`. Sendo assim, se `VarQuant` é igual a 1, apenas `VarIni` é lida do controlador. Se `VarQuant` é igual a 2 serão lidas do controlador o conteúdo de `VarIni` e `VarIni+1`, e assim, sucessivamente.
Caso a quantidade especificada ultrapasse o número de variáveis definidas no controlador, será retornado pela função um código de falha indicando esta situação.

Parâmetros de Saída

`DataBuf` Buffer do usuário onde serão salvos os valores obtidos pelo *driver* de comunicação. É de responsabilidade de o usuário prover o *buffer* com o espaço necessário para receber todos os dados solicitados. Não é realizado nenhum teste de consistência do tamanho do *buffer*. O tamanho mínimo do *buffer* deverá ser:

Tbuf(min) = `VarQuant` * Tamanho de `VarType`

Código de Retorno

`= 0 (SUCCESS)` Indica função executada com sucesso.
`<> 0` Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090.

`Notas` Apenas alguns modelos de controladores (equipados com firmware G3 ou superior) possuem suporte para variáveis do tipo L.

6.8 SCPWriteData

Função 08 **SCPWriteData**

Protótipo da função em C/C++

```
int WINAPI SCPWriteData (client_device_id, int TypeVar, int VarIni, int VarQuant, void* DataBuf)
```

Descrição

Transfere para o controlador o conteúdo do *buffer* de variáveis especificadas na função. O usuário deve especificar o tipo de variável (R, M, D ou L), identificar o número da variável inicial e a quantidade de variáveis a serem transferidas a partir desta.

Cada tipo de variável requer um espaço distinto no *buffer* de transmissão conforme descrito na tabela a seguir:

Tipo da Variável	Tipo associado na linguagem C	Tamanho
R	*unsigned char*	1 byte (8 bits)
M	*short int*	2 bytes (16 bits)
D	*float*	4 bytes (32 bits)
L	*int*	4 bytes (32 bits)

O tamanho máximo de um *frame* de dados no protocolo SCP-HI é de 249 bytes. Isto define a quantidade máxima de variáveis de cada tipo que pode ser transferida para o controlador em um único comando. Quando a quantidade de variáveis a ser enviada ultrapassar este limite, a função divide o buffer de transmissão, enviando tantos *frames* de comunicação quanto forem necessários para transferir todos os dados especificados.

Parâmetros de Entrada

`Client_device_id` Identificador do equipamento pela aplicação. Este nro deve compor o identificador do *driver* do servidor associado ao equipamento a ser acessado e o endereço do equipamento.
Por exemplo, para acessar o controlador com endereço 20 localizado no *driver* 3000, utilize para este parâmetro o valor 3020. Note que o canal do *driver* associado no SCP Server II deve remapear este valor para o endereço corrente, ou seja, o endereço 3020 deve ser remapeado para 20.
Se for utilizado um valor abaixo de 1000, a DLL considera que este endereço está mapeado no *driver* 1000 e encaminha o frame associado para este *driver*.
(Vide capítulo 4 para esclarecimentos adicionais)

`TypeVar` Identifica o tipo de variável a ser solicitada do controlador, conforme tabela a seguir:

TypeVar	Tipo de variável	ID	Tamanho	Faixa de valores	
				Min	Max
0	Contato auxiliar	R	1 byte	OFF(0)	ON(255)
1	Memórias inteiras	M	2 bytes	-32768	+32767
2	Memórias reais	D	4 bytes	-10E-38	+10E+38
3	Memórias long	L	4 bytes	-2147483648	+2147483647

`VarIni`	Identifica o número da variável inicial a ser transferida. Todas as variáveis da base de dados dos controladores HI começam em zero (0) e vão até o número máximo de variáveis do tipo – 1. O número de variáveis de cada tipo depende do tipo de firmware carregado no controlador e, no caso de firmware de PLC este número é alocado dinamicamente em função do programa de aplicação criado. Caso seja especificada uma variável não definida no controlador, será retornado pela função um código de falha, indicando esta situação.
`VarQuant`	Especifica o número de variáveis a serem transferidas a partir de `VarIni`. Sendo assim, se `VarQuant` é igual a 1, apenas `VarIni` é enviada para o controlador. Se `VarQuant` é igual a 2 serão transferidas para do controlador o conteúdo de `VarIni` e `VarIni+1`, e assim sucessivamente. Caso a quantidade de variáveis especificada ultrapasse o número de variáveis definidas no controlador, será retornado pela função um código de falha indicando esta situação.
`DataBuf`	*Buffer* do usuário com o vetor de valores do tipo especificado a serem transferidos para o controlador.

`Parâmetros de Saída`

`Não se aplica`

`Código de Retorno`

`= 0 (SUCCESS)`	Indica função executada com sucesso.
`<> 0`	Indica condição de falha. Para maiores informações sobre os códigos de retorno consultar ENA.00090.

`Notas` Apenas alguns modelos de controladores (equipados com firmware G3 ou superior) possuem suporte para variáveis do tipo L.

7. Definições do protocolo

Estas constantes estão definidas no arquivo HI_Defines.h, fornecido juntamente com a DLL de comunicação.

```
// ____ Definições globais

SUCCESS                     0        // código de sucesso
FAIL                        -1       // código geral de falha

PLC_GLOBAL_STATION          255      // Endereço de comunicação global


// ____ Tipos de variáveis dos Controladores HI

PLC_R_TYPE                  0        // variável do tipo R
PLC_M_TYPE                  1        // variável do tipo M
PLC_D_TYPE                  2        // variável do tipo D
PLC_L_TYPE                  3        // variável do tipo L
```

8. Códigos de retorno

Os códigos de retorno possíveis de serem obtidos pelo protocolo de comunicação: Códigos de Retorno do Controlador e Códigos de Retorno de Aplicativos estão disponíveis para consulta na ENA.00090

8.1 Referências

- ENA.00090 Lista de Códigos de Retorno de Funções.

Obs.: Este documento encontra-se disponível para *download* em nosso *site*:
www.hitecnologia.com.br

Controle do Documento

Considerações gerais

1. Este documento é dinâmico, estando sujeito a revisões, comentários e sugestões. Toda e qualquer sugestão para seu aprimoramento deve ser encaminhada ao departamento de suporte ao cliente da **HI Tecnologia**, especificado na "Apresentação" deste documento.

2. Os direitos autorais deste documento são de propriedade da **HI Tecnologia.**

Responsabilidades pelo documento

	Data	Responsável	
Elaboração	12/01/2011	Helio J. Almeida Jr.	
Revisão	30/10/2012	Heber A. Scachetti	*Revisado em mídia*
Aprovação	30/10/2012	Heber A. Scachetti	*Aprovado em mídia*

Histórico de Revisões

Data	Rev	Descrição
30/10/2012	2	Atualização dos códigos de erros, transferência funções sistema arquivo para PNS.00034
29/05/2012	1	Adição das funções de configuração da apresentação de dados de data-hora
14/01/2011	0	Documento original