Manual Funções DLL Companytec

Manual

DT433

Leitura de Cartão

Comandos de Registro de Cartão na Memória

Comandos de Gerencia de Lista Negra

Predeterminação Identificada

Comandos de Log/Depuração

Comandos de Comunicação Via Protocolo Companytec

Introdução:

Caro desenvolvedor. Essa biblioteca foi desenvolvida para facilitar a implementação dos equipamentos Companytec em seu software. Disponibilizando todas as funções necessárias para a integração.

Esta DLL foi desenvolvida especificamente para integração do sistema com o equipamento CBC e Horustech, disponibilizando funções de suporte que permitem que o desenvolvedor possa se comunicar com o concentrador companytec, para busca de informações diretamente relacionadas com a bomba e com o sensor Identfid.

A DLL Companytec foi criada utilizando a linguagem C, utilizando o convenção para chamada de função CDCEL. Caso nenhuma função da DLL seja chamada com sucesso, talvez sua linguagem utilize outro padrão para chamada (como exemplo STDCALL) sendo necessário que se explicite o tipo de chamada no protótipo da função.

As funções da DLL quando utilizam dados do tipo caractere, sejam eles um único caractere ou múltiplos (formando assim ***strings*** de caractere), possuem na sua nomenclatura o sufixo 'A' ou 'W' como segue exemplo abaixo.

```
int     OpenSocketA      ( char* IP);
int     OpenSocketW ( wchar_t* IP);
```

As funções com final 'A' utilizam caractere da família ANSI com 8 bits para cada caractere. As funções com final 'W' utilizam caractere da família UNICODE (ou WIDE) com 16bits para cada caractere. Funções que não utilizam o tipo caractere não possuem a nomenclatura com final 'A' ou 'W'.

As informações retornadas nas funções da DLL terão seus modelos descritos para cada função. Quando houver algum erro na comunicação da DLL com o concentrador ou na execução da DLL, mensagens de erro foram criadas para descrevê-las.

As mensagens de erro são divididas em mensagens com retorno do tipo caractere e retorno com tipos numéricos.

- Tipo Caractere:

- #ERROR;-1 Mensagem de erro de conexão
- #ERROR;-2 Mensagem de erro de parâmetros de entrada, este caso se dá quando os parâmetros de passagem não estão no formato correto para que possam ser processados de tal forma que haja efetiva comunicação com o concentrador.
- #ERROR;-3 Mensagem de erro do tipo TIMEOUT, não houve no tempo determinado para cada função resposta do concentrador.

- Tipo numérico:

- -1 Mensagem de erro de conexão
- -2 Mensagem de erro de parâmetros de entrada, este caso se dá quando os parâmetros de passagem não estão no formato correto para que possam ser processados de tal forma que haja efetiva comunicação com o concentrador.
- -3 Mensagem de erro do tipo TIMEOUT, não houve no tempo determinado para cada função resposta do concentrador.

As funções da DLL quando apresentam retorno por tipo caractere, os blocos de informação são divididos pelo separador ';', no exemplo abaixo a string de retorno devolve o total a pagar, o volume abastecido e o código de virgula, logo a string segue o modelo apresentado no exemplo abaixo:

123,321;456,65;0C

Então ao analisar esta string de retorno temos que:

Total a pagar = 123,321
Volume abastecido = 456,65
Código de Bico = 0C

Nos comandos apresentados abaixo cada bloco de informação vai ter um tipo relacionado para que fique mais fácil a interpretação da string e para que seja mais fácil manipular os dados.

No caso acima temos:

Total a pagar = float
Volume abastecido = float
Código de bico = Byte (hexadecimal)

Comunicação:

A interface de comunicação com o concentrador, utiliza a comunicação TCP na porta 2001 ou serial. A comunicação deve ser iniciada com a chamada " **OpenSocket** " ou "**OpenSerial**".

O inicio de comunicação com uma interface não implica em comunicação com a outra interface.

Interface Concentrador

int OpenSocketA (char* IP); **int OpenSocketW (wchar_t* IP);**	
Função que abre a porta de comunicações para envio e recebimento de comandos. Essa função só necessita ser chamada uma vez, no inicio da aplicação que irá comunicar com nosso equipamento.	
Parâmetros de entrada:	
• IP: Endereço na rede em que será feito a conexão	char*/ wchar_t*
Retorno da função:	
• 1 caso sucesso ou 0 caso falha	

int CloseSocket ( );	
Função chamada para finalizar uma conexão do tipo Ethernet	
Parâmetros de entrada:	
• Nenhum	
Retorno da função:	
• 1 em caso de sucesso ou 0 em caso de falha	

int OpenSerial(int port)	
Função que abre a porta de comunicações para envio e recebimento de comandos. Essa função só necessita ser chamada uma vez, no inicio da aplicação que irá comunicar com nosso equipamento.	
Parâmetros de entrada:	
• Port: número da porta para abrir comunicação serial	Int
Retorno da função:	
• 1 caso sucesso ou 0 caso falha	

int CloseSerial (int port)	
Função chamada para finalizar uma conexão do tipo Ethernet	
Parâmetros de entrada:	
• Nenhum	
Retorno da função:	
• 1 em caso de sucesso ou 0 em caso de falha	

char* GetSaleA ( );
wchar_t* GetSaleW ( );

Ler o abastecimento atual em memória.

Parâmetros de entrada:

- Nenhum

Retorno da função:

A função pode ter 3 retornos possíveis,

- Sucesso
- ‘0’ caso não haja qualquer abastecimento a ser lido.
- #ERROR.

Para caso de sucesso os campos são definidos como segue abaixo.

TTTTTT;LLLLLL;PPPP;CCCC;BB;DD/MM;HH:mm;RRRR;EEEEEEEEEE

- TTTTTT: Total a Pagar; (Float) (bombas mecânicas retornam “000000”);
- LLLLLL: Volume abastecido (Litros); (Float)
- PPPP: Preço unitário; (Float)
- CCCC: Tempo de abastecimento; (inteiro em segundos)
- BB: Código de bico; (Byte representado em hexadecimal)
- DD/MM: Data (dia/mes);
- HH/mm: Hora; (hora:minuto)
- RRRR: Número do abastecimento; (inteiro)
- EEEEEEEEEE: Encerrante do bico (float com duas casas decimais);

- Exemplo:

0003,70;000,366;9,999;0056;0C;22/03;23:46;1544;00032845,68

char* GetSalePafA ( );
wchar_t* GetSalePafW ( );

Utilizado para ler os abastecimentos com identificação de cliente e frentista.

Parâmetros de entrada:

- Nenhum

Retorno da função:

A função pode ter 3 retornos possíveis,

- Sucesso
- '0' caso não haja qualquer abastecimento a ser lido.
- #ERROR.

Para caso de sucesso os campos são definidos como segue abaixo.

TTTTTT;LLLLLL;PPPP;CCCC;BB;DD/MM;HH:mm;RRRR;FFFFFFFFFF;IIIIIIIIII;ffffffffffffffff;kkkkkkkkkkkkkkkk

- TTTTTT: Total a Pagar; (Float) (bombas mecânicas retornam "000000");
- LLLLLL: Volume abastecido (Litros); (Float)
- PPPP: Preço unitário; (Float)
- CCCC: Tempo de abastecimento; (inteiro em segundos)
- BB: Código de bico; (Byte representado em hexadecimal)
- DD/MM: Data (dia/mês);
- HH/mm: Hora; (hora:minuto)
- RRRR: Número do abastecimento; (inteiro)
- FFFFFFFFFF : Encerrante Final (Float com duas casas decimais);
- IIIIIIIIII : Encerrante Inicial (Float com duas casas decimais);
- yyyyyyyyyyyyyyyy : Tag Frentista
- kkkkkkkkkkkkkkkk : Tag Cliente

- Exemplo sem cartão:
0003,70;000,366;9,999;0056;0C;22/03;23:46;001544;00032845,68;00032845,31;FFFFFFFFFFFFFFFF;0000000000000000
- Exemplo com cartão frentista:
0003,70;000,366;9,999;0056;0C;22/03;23:46;001544;00032845,68;00032845,31;B312345678912345;0000000000000000

char* GetSaleTWCA (VOID);
wchar_t* GetSaleTWCW (VOID);

Utilizado para ler os abastecimentos com identificação de cliente e frentista através do terminal.

Parâmetros de entrada:

- Nenhum

Retorno da função:

A função pode ter 3 retornos possíveis,

- Sucesso
- '0' caso não haja qualquer abastecimento a ser lido.
- #ERROR.

Para caso de sucesso os campos são definidos como segue abaixo.

TTTTTT;LLLLLL;PPPP;CCCC;BB;DD/MM;HH:mm;RRRR;FFFFFFFFFF;IIIIIIIIII;ffffffffffffffff;kkkkkkkkkkkkkkkk;bb;cc;tt;oooooooo

- TTTTTT: Total a Pagar; (Float) (bombas mecânicas retornam "000000");
- LLLLLL: Volume abastecido (Litros); (Float)
- PPPP: Preço unitário; (Float)
- CCCC: Tempo de abastecimento; (inteiro em segundos)
- BB: Código de bico; (Byte representado em hexadecimal)
- DD/MM: Data (dia/mês);
- HH/mm: Hora; (hora:minuto)
- RRRR: Número do abastecimento; (inteiro)
- FFFFFFFFFF : Encerrante Final (Float com duas casas decimais);
- IIIIIIIIII : Encerrante Inicial (Float com duas casas decimais);
- yyyyyyyyyyyyyyyy : Tag Frentista (string)
- kkkkkkkkkkkkkkkk : Tag Cliente (string)
- bb : número do bico na pista (inteiro)
- cc : tipo de combustível (inteiro)
- tt : número do tanque (inteiro)
- 00000000 : ôdometro/horímetro (string)

- Exemplo sem cartão:
0003,70;000,366;9,999;0056;0C;22/03;23:46;001544;00032845,68;00032845,31;FFFFFFFFFFFFFFFF;0000000000000000
- Exemplo com cartão frentista:
0003,70;000,366;9,999;0056;0C;22/03;23:46;001544;00032845,68;00032845,31;B312345678912345;0000000000000000

VOID NextSale ( );

Função que tem por finalidade informar ao concentrador que o abastecimento atual já foi lido e armazenado. Quando este comando é enviado, a fila de abastecimentos na memoria é incrementada passando o registro para o próximo abastecimento válido.

Parâmetros de entrada:

- Nenhum

Retorno da função:

- Nenhum

char* VisualizeA ( );
wchar_t* VisualizeW ( );

Comando para visualizar os abastecimentos em andamento. Este comando não deve ser utilizado para qualquer tipo de verificação ou validação no seu sistema, dado que os valores apresentados tem atraso relacionado ao tempo de processamento da bomba e do concentrador.

Parâmetros de entrada:

- Nenhum

Retorno da função:

A função pode ter 3 retornos possíveis,

- Sucesso
- '0' caso não haja qualquer abastecimento a ser lido.
- #ERROR.

Em caso de sucesso, a string de retorno informa para cada bloco de informação o código de bico responsável pelo abastecimento seguido de um caractere '|' e o valor do abastecimento.
O modelo para cada bloco é:

CC|TTTTTT

- CC: código de bico (Byte com representação em Hexadecimal)
- TTTTTT: Valor do Abastecimento (Float)

- Exemplo:

08|0002,74;09|0001,04;0C|0024,60

Bicos abastecendo : 08 , 09 , 0C

char* GetStatusA ( );
wchar_t* GetStatusW ( );

Comando utilizado para ler a situação de cada bomba conectada ao equipamento.

Parâmetros de entrada:

• Nenhum	

Retorno da função:

A função pode ter 2 retornos possíveis,

- Sucesso
- #ERROR.

Em caso de sucesso a string de retorno, possuirá 32 caracteres separados pelo caractere ';'. Cada caractere representa o status de um endereço, os status possíveis são.

- L Bomba encontra-se livre para abastecer.
- B Bomba bloqueada para realizar abastecimentos
- C Bomba concluiu abastecimento
- A Bomba está em processo de abastecimento
- E Bomba está aguardando liberação da automação para iniciar o processo de abastecimento.
- F Bomba não presente ou em falha
- P Bomba está pronta para abastecer

Exemplo:

F;F;F;F;A;A;F;F;A;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F;F

int FreePump (unsigned char codbico);

Libera bomba para abastecimentos.

Parâmetros de entrada:

• codbico: código de bico que deseja liberar a bomba	unsigned char/ Byte

Retorno da função:

A função pode ter 3 retornos possíveis,

- 1 em caso de sucesso
- 0 em caso de falha
- #ERROR.

int BlockPump (unsigned char codbico);	
Bloqueia bomba para abastecimentos	
Parâmetros de entrada:	
• codibico: código de bico que deseja liberar a bomba	Unsigned char/ Byte
Retorno da função:	
A função pode ter 3 retornos possíveis, • 1 em caso de sucesso • 0 em caso de falha • #ERROR.	

int StopPump (unsigned char codbico);	
Parar Abastecimento (não implementado em todas as bombas)	
Parâmetros de entrada:	
• codbico: código de bico que deseja liberar a bomba	Unsigned char/ Byte
Retorno da função:	
A função pode ter 3 retornos possíveis, • 1 em caso de sucesso • 0 em caso de falha • #ERROR.	

int AutoPump (unsigned char codbico);	
Autoriza a bomba para realizar um abastecimento. Após finalizar o abastecimento, a bomba voltará para o status anterior a autorização.	
Parâmetros de entrada:	
• codbico: código de bico que deseja liberar a bomba	Unsigned char/ Byte

Retorno da função:

A função pode ter 3 retornos possíveis,

- 1 em caso de sucesso
- 0 em caso de falha
- #ERROR.

int PausePump (unsigned char codbico);

Pausa um abastecimento, o abastecimento poderá ser retomado com este comando.

Parâmetros de entrada:

• codbico: código de bico que deseja liberar a bomba	Unsigned char/ Byte

Retorno da função:

A função pode ter 3 retornos possíveis,

- 1 em caso de sucesso
- 0 em caso de falha
- #ERROR.

int SetPumpPrice (unsigned char codbico , float price);

Comando utilizado para alterar o preço num determinado bico.

Parâmetros de entrada:

• codbico: código de bico • price: valor a ser predeterminado price deve ter 3 casas decimais, com valor máximo permitido de 9,999.	Unsigned char/ Byte Float

Retorno da função:

A função pode ter 3 retornos possíveis,

- 1 em caso de sucesso
- 0 em caso de falha
- -1 em erro de conexão
- -2 em erro de parâmetro de entrada

int PresetPump(unsigned char codbico , int value);	
Comando utilizado para determinar o valor do abastecimento.	
Parâmetros de entrada:	
• codbico : Código de bico; • value: valor a ser predeterminado, o valor em reais deve ser multiplicado por mil. Exemplo: Caso queira enviar o preset de 50 reais deve ser enviado o inteiro 5000. Exemplo: Caso queira enviar o preset de 30,50 reais deve ser enviado o inteiro 3050.	Unsigned char/Byte int
Retorno da função:	
A função pode ter 3 retornos possíveis, • 1 em caso de sucesso • 0 em caso de falha • -1 em erro de conexão • -2 em erro de parâmetro de entrada	

float GetTotalValue(unsigned char codbico);	
Devolve o totalizador em valor do bico.	
Parâmetros de entrada:	
• Codbico: Código de bico	
Retorno da função:	
• Valor maior que 0 . Sucesso • -1 erro de conexão • -3 erro timeout	

float GetTotalVolume (unsigned char codbico);	
Devolve o totalizador em volume do bico.	
Parâmetros de entrada:	
• Codbico: Código de bico	
Retorno da função:	
• Valor maior que 0 . Sucesso • -1 erro de conexão • -3 erro timeout	

float GetTotalVolume (unsigned char codbico);	
Devolve o totalizador em volume com a casa do milhão do bico.	
Parâmetros de entrada:	
• Codbico: Código de bico	
Retorno da função:	
• Valor maior que 0 . Sucesso • -1 erro de conexão • -3 erro timeout	

char* GetClockA ( ); **wchar_t* GetClockW ( );**
Retorna o relógio da automação
Parâmetros de entrada:
• Nenhum
Retorno da função:
DD/MM/AA;HH/mm
Exemplo: 24/03/15;14:28

int SyncClock (VOID);

Sincroniza o relógio do concentrador com o do computador.

Parâmetros de entrada:

- Nenhum

Retorno da função:

- 1 em caso de Sucesso
- 0 em caso de falha

char* GetIdentTagA ( );
wchar_t* GetIdentTagW ( );

Caso o identificador lido pelo sensor não esteja cadastrado na memória do IdentFid II, essa informação será enviada ao PC (quando o mesmo solicitar), que por sua vez, autorizará ou não a bomba a abastecer, mediante envio de comando para a automação.

Parâmetros de entrada:

- Nenhum

Retorno da função:

- Sucesso
- 0 em caso de falha
- #ERROR

Em caso de Sucesso a string de retorno segue o padrão:

TTTTTTTTTTTTTTTT;CC;DD/MM;HH:mm

- TTTTTTTTTTTTTTTT: tag
- CC: Código de Bico
- DD/MM: Data
- HH:mm: Hora

- Exemplo:

B3CF6CC7B739B03C;04;24/03;15:13

VOID NextIdentTag (VOID);

Incrementa o registro de cartões de identFid, na memória de cartões que forma passados e não estavam registrados na memória de cartões do concentrador.

Parâmetros de entrada:

• Nenhum	

Retorno da função:

- Nenhum

char* GetIdentTagFromMemoryIdxA (int idx);
wchar_t* GetIdentTagFromMemoryIdxW (int idx);

Le um cartão armazenado na memória de cartões do concentrador no índice expecificado.

Parâmetros de entrada:

• Index	int

Retorno da função:

- Sucesso
- '0' em falha
- #ERROR

Em caso de Sucesso a string de retorno tem o seguinte format:

TTTTTTTTTTTTTTT;PP;NNNNNN;MMMMMM

- TTTTTTTTTTTTTTT: Tag
- PP: Permissão
  - 04 – cliente sem permissão para liberar a bomba
  - 24 - cliente com permissão para liberar a bomba
  - 27 - frentista com permissão para liberar a bomba
- NNNNNN: posição do cartão
- MMMMMM: numero de cartões na memória

- Exemplo:

B3CF6CA5B7EC8B19;27;000000;000002

int InsertIdentTagInMemoryA (int CodId , char* tag);
int InsertIdentTagInMemoryW (int CodId , wchar_t* tag);

Insere um cartão IdentFid na memória do concentrador.

Parâmetros de entrada:

• codId - código de identificação de cliente ○ 04 – cliente sem permissão para liberar a bomba ○ 24 - cliente com permissão para liberar a bomba ○ 27 - frentista com permissão para liberar a bomba • Tag - tag de identificação	Int caractere

Retorno da função:

- Retorno maior que 0, Sucesso
- -1 caso erro de conexão
- -2 erro de parametro de entrada
- -3 erro de timeout

int DeleteIdentTagInMemoryA (int CodId , char* tag);
int DeleteIdentTagInMemoryW (int CodId , wchar_t* tag);

Deleta um cartão da memória do concentrador

Parâmetros de entrada:

• codId - código de identificação de cliente ○ 04 – cliente sem permissão para liberar a bomba ○ 24 - cliente com permissão para liberar a bomba ○ 27 - frentista com permissão para liberar a bomba • Tag - tag de identificação	Int caractere

Retorno da função:

- Retorno maior que 0, Sucesso
- -1 caso erro de conexão
- -2 erro de parametro de entrada
- -3 erro de timeout

VOID EraseIdentTagMemory (VOID);	
Apaga a memória de cartões do concentrador.	
Parâmetros de entrada:	
• Nenhum	
Retorno da função:	
• Nenhum	

int PutIdentTagInBlackListA (char* tag); **int PutIdentTagInBlackListW (wchar_t* tag);**	
Coloca a tag de identificação na lista negra de cartões.	
Parâmetros de entrada:	
• Tag	Caractere
Retorno da função:	
• 1 em caso de sucesso • 0 em caso de falha • -1 em caso de erro de conexão • -2 em casso de erro de parâmetro de entrada • -3 em caso de erro de timeout	

int TakeIdentTagOutBlackListA (char* tag); **int TakeIdentTagOutBlackListW (wchar_t* tag);**	
Retira a tag de identificação na lista negra de cartões.	
Parâmetros de entrada:	
• Tag	Caractere
Retorno da função:	
• 1 em caso de sucesso • 0 em caso de falha • -1 em caso de erro de conexão • -2 em casso de erro de parâmetro de entrada • -3 em caso de erro de timeout	

int ClearIdentTagBlackList (VOID);	
Limpa a lista negra de cartões.	
Parâmetros de entrada:	
• Nenhum	
Retorno da função:	
• 1 em caso de sucesso • 0 em caso de falha • -1 em caso de erro de conexão • -2 em casso de erro de parâmetro de entrada • -3 em caso de erro de timeout	

int PresetPumpPriceIdfA (unsigned char codBico , char* tag, int CodId , char autoriza , float price);

int PresetPumpPriceIdfW (unsigned char codBico , wchar_t* tag, int CodId , wchar_t autoriza , float price);

Predetermina um valor para o abastecimento com informação de identificação. A tag do cartão passado neste método não necessita estar gravado na memoria do concentrador.

Parâmetros de entrada:

- Codbico: código de bico.
- Tag: código de identificação.
- codId: tipo de tag identFid. (0 código de frentista 1 cliente)
- Autoriaza: 'S' sim 'N' não autoriza a bomba para abastecer.
- Price: Valor para a predeterminação.

Retorno da função:

- 1 em caso de sucesso
- 0 em caso de falha
- -1 em caso de erro de conexão
- -2 em casso de erro de parâmetro de entrada
- -3 em caso de erro de timeout

char* GetLastPacketSendedA (VOID);
wchar_t* GetLastPacketSendedW (VOID);

Retorna o Ultimo pacote enviado para o concentrador no formato do protocolo Companytec.

Parâmetros de entrada:

- Nenhum

Retorno da função:

- String no format do protocol Companytec.

char* GetLastPAcketReceivedA (VOID); **wchar_t* GetLastPAcketReceivedW (VOID);**	
Retorna o Ultimo pacote retornado do concentrador no formato do protocolo Companytec.	
Parâmetros de entrada:	
• Nenhum	
Retorno da função:	
• String no format do protocol Companytec.	

char* SendReceiveA (char* str , int timeout); **wchar_t* SendReceiveW (wchar_t* str , int timeout);**	
Realiza a comunicação com o concentrador utilizando os comandos no formato Companytec.	
Parâmetros de entrada:	
• Str : string do comando no formato Companytec • Timeout: tempo para esperar uma resposta do concentrador(especificado no protocolo de comunicação Companytec)	Caractere int
Retorno da função:	
• String no format do protocol Companytec.	

Companytec Automação e Controle Ltda.

Av. Ferreira Viana, 1421 - Areal - 96080-000 - Pelotas - RS

www.companytec.com.br

Fone: (53) 3284-8100

suporte@companytec.com.br