

Driver ASCII Generic

Nome do Arquivo	ASCII.dll
Fabricante	Elipse Software
Equipamentos	Qualquer equipamento com interface ASCII de comunicação
Protocolo	Qualquer protocolo ASCII
Versão	2.0.5
Última Atualização	10/08/2026
Plataforma	Win32
Dependências	IOKit versão 2.0 ou superior
Leitura de Superblocos	Não
Nível	0

Introdução

O Driver ASCII Generic comunica-se com qualquer equipamento que utilize o formato **ASCII**.

Configuração do Driver

Os parâmetros **[P]** de configuração deste Driver não são utilizados. Todas as configurações devem ser realizadas na caixa de diálogo de configurações deste Driver. Para mais informações sobre as abas **Setup**, **Serial**, **Ethernet**, **Modem** e **RAS**, consulte o tópico Documentação das Interfaces de Comunicação.

Configuração das Propriedades

A aba **ASCII** contém configurações específicas para este Driver. As opções disponíveis nesta aba estão descritas na tabela a seguir.

Opções disponíveis na aba ASCII

OPÇÃO	DESCRIÇÃO
Max Answer Bytes	Número máximo de bytes aguardados na resposta

OPÇÃO	DESCRIÇÃO
Max Answer Elements	Número máximo de variáveis, ou elementos, em um comando ou resposta
ID	Índice do comando associado ao parâmetro <i>N2/B2</i> dos Tags PLC ou Bloco
TX	Formato do comando de envio. Para mais informações, consulte o tópico Configuração dos Comandos ASCII
RX	Formato do comando de recepção. Para mais informações, consulte o tópico Configuração dos Comandos ASCII

As opções **ID**, **TX** e **RX** são itens que compõem uma listagem de comandos definidos pelo usuário. Preencha estas opções de um comando e adicione-o na listagem de comandos usando a opção **Add**. Para alterá-los, selecione um comando na listagem e utilize a opção **Update** para modificá-lo ou **Del** para excluí-lo.

Configuração dos Comandos ASCII

Os comandos ASCII são montados com o objetivo de enviar ou extrair dados dos pacotes de comunicação em conformidade com o protocolo de um equipamento. Para cada formato de envio e de recepção, associa-se um índice, ou ID. No comando podem existir quaisquer números ou letras em formato **ASCII**, fixos ou com campos variáveis. Para criar campos variáveis, a regra geral é utilizar o formato **%Nt**, conforme descrito na tabela a seguir.

Formatos disponíveis para comandos ASCII

FORMATO	DESCRIÇÃO
%	Símbolo identificador de campo variável. Caso a mensagem precise enviar um caractere % (porcentagem), utilize o valor "%%"
N (opcional)	Número de caracteres ou formatador. Se o campo é um número real, N pode ser um número real, como por exemplo 5.2, indicando cinco ou mais algarismos na parte inteira e mais dois algarismos na parte fracionária
t	Tipo de dados. Os valores possíveis são d : Decimal com sinal, u : Decimal sem sinal, x : Hexadecimal, f : Número real, b : Binário (valor binário do caractere), s : String (caracteres de qualquer tipo), t : Texto (somente caracteres que podem ser impressos, ou seja, cujo código ASCII é maior ou igual a 32 ou 20h) ou n : Interpreta o valor decimal no formato BCD

NOTA

Para os tipos de dados **s** (**String**) e **t** (texto) existem os subtipos **@** e **#**, que podem ser utilizados quando se deseja extrair somente uma porção de uma **String** ou de um texto:

- Utilize o símbolo **@** para identificar um caractere especial como delimitador inicial da **String**
- Utilize o símbolo **#** para identificar um caractere especial como delimitador final da **String**. Também é possível definir este caractere especial com notação hexadecimal, utilizando o símbolo **h** seguido de mais dois caracteres que representam o código ASCII em hexadecimal
- Utilize o formato **%s** para extrair uma **String** completa
- Utilize o formato **%@g#;s** para extrair uma *substring* do caractere **g** até o caractere **;** (ponto e vírgula)
- Utilize o formato **%@h02#h03s** para extrair uma *substring* do caractere **2** até o caractere **3**
- Os caracteres delimitadores dos símbolos **@** e **#** não são parte da extração dos dados que representam a *substring* do formato **%s**

Podem ser inseridos ou verificados também os caracteres especiais **lr** (*carriage return*), **ln** (*line feed*) e qualquer outro caractere especial no formato **/hXX**, inserindo os caracteres **/** (barra) ou **** (barra invertida) seguidos de **r**, **n** ou **h** diretamente na mensagem **TX** ou **RX**. A inserção ou comparação é realizada diretamente com os caracteres especiais **lr** (13), **ln** (10) ou **/hXX**, onde **XX** representa um caractere especial em formato hexadecimal de dois dígitos.

Também é possível definir qual é o byte inicializador da recepção dos dados, qual é o byte terminador da recepção dos dados e se estes bytes devem ou não ser parte da extração dos dados, conforme as opções listadas a seguir:

- **[xx:** Define qual é o byte que inicializa a recepção dos dados, em que **xx** está em notação hexadecimal e está incluído na extração dos dados
- **]xx:** Define qual é o byte que termina a recepção dos dados, em que **xx** está em notação hexadecimal e está incluído na extração dos dados
- **(xx:** Define qual é o byte que inicializa a recepção dos dados, em que **xx** está em notação hexadecimal e não está incluído na extração dos dados
- **)xx:** Define qual é o byte que termina a recepção dos dados, em que **xx** está em notação hexadecimal e não está incluído na extração dos dados

Adicionalmente, para as respostas **RX** que utilizem os bytes terminadores **]xx** ou **)xx** e se houver a necessidade de descartar alguns bytes que vêm logo a seguir, é possível acrescentar um parâmetro para indicar a quantidade de bytes que devem ser descartados por este Driver após ter recebido este byte terminador, com o objetivo prático de efetuar uma limpeza do *buffer* de recepção dos dados.

Para utilizar o parâmetro de descartar os últimos *n* bytes, utilize a expressão **"*nn"** logo após o parâmetro que define o byte terminador de recepção de dados, ou seja, **]xx*nn** ou **)xx*nn**, em que **nn** está em notação decimal e representa a quantidade de bytes que são recebidos e descartados por este Driver.

As tabelas a seguir contêm exemplos de utilização de comandos ASCII.

Exemplo de utilização

Protocolo	Objetivo	Configuração

TX: <STX>query<ETX> RX: <STX>1200<ETX>	Enviar o comando query a um equipamento e extrair o valor numérico da resposta	• ID: 1 (um) • TX: /h02query/h03 • RX: (02%u)03 • N1/B1: 0 (zero, envia e recebe comando) • N2/B2: 1 (um, ID do comando) • Read Tag: 1200
---	---	--

O pacote **TX** é montado para enviar o comando **query** conforme a estrutura do protocolo de um equipamento, ou seja, utilizando "/h02" (byte STX 0x02) + "query" + "/h03" (byte ETX 0x03).

O pacote **RX** é montado com o formato "(02" (inicializador não incluso STX 0x02) + "%u" + ")03" (terminador não incluso ETX 0x03), em que o parâmetro %u extrai os dados no formato numérico sem sinal.

Exemplo de utilização

Protocolo	Objetivo	Configuração
TX: \$memory13<CR> <LF> RX: \$34.45<CR><LF>	Enviar o comando memory mais a posição de memória a um equipamento e extrair o valor numérico da resposta	• ID: 1 (um) • TX: \$memory%u/r/n • RX: (24%f)0D*1 • N1/B1: 1 (um, valores TX) • N2/B2: 1 (um, ID do comando) • Write Tag: 13 • N1/B1: 0 (zero, envia e recebe comando) • N2/B2: 1 (um, ID do comando) • Read Tag: 34.45

O pacote **TX** requer a configuração de uma posição de memória. Para isto é montado o pacote **TX** com a inserção de um campo variável %u, a ser definido via escrita do Tag com o parâmetro *N1/B1* igual a 1 (um, valores **TX**). Após preencher o campo variável %u com um valor numérico, neste exemplo o valor 13, pode-se finalmente transmitir o pacote **TX** ao equipamento no formato "\$" + "memory" + "13" + /r (byte CR 0x0D) + /n (byte LF 0x0A).

O pacote **RX** é montado com o formato "(24" (inicializador não incluso "\$") + "%f" + ")0D*1" (terminador não incluso CR 0x0D + descarte do byte LF 0x0A que vem a seguir), em que o parâmetro %f extrai os dados no formato numérico real.

Exemplo de utilização

Protocolo	Objetivo	Configuração

TX: #GetData<CR> <LF> RX: #12;452;75<CR> <LF>	Enviar o comando #GetData a um equipamento e extrair três valores numéricos da resposta	• ID: 1 (um) • TX: #GetData/r/n • RX: (23%u;%u;%u)0D*1 • N1/B1: 0 (zero, envia e recebe comando) • N2/B2: 1 (um, ID do comando) • Read TagBloco: • Elemento 1: 12 • Elemento 2: 452 • Elemento 3: 75
--	--	---

O pacote **TX** é montado para enviar o comando **GetData** conforme a estrutura do protocolo de um equipamento, ou seja, utilizando "# + "GetData" + /r (byte CR 0x0D) + /n (byte LF 0x0A).

O pacote **RX** é montado com o formato "(23" (inicializador não incluso "#") + "%u" + ";" + "%u" + ";" + "%u" + ")0D*1" (terminador não incluso CR 0x0D + descarte do byte LF 0x0A que vem a seguir), em que são extraídos os três formatos %u através de um Tag Bloco com três Elementos, um para cada formato %u.

Exemplo de utilização

Protocolo	Objetivo	Configuração
TX: <STX>DeviceName<ETX> RX: <STX>Personal Computer <ETX>	Enviar o comando DeviceName a um equipamento e extrair o valor no formato String da resposta	• ID: 1 (um) • TX: /h02DeviceName/h03 • RX: (02%s)03 • N1/B1: 0 (zero, envia e recebe comando) • N2/B2: 1 (um, ID do comando) • Read Tag: "Personal Computer"

O pacote **TX** é montado para enviar o comando **DeviceName** conforme a estrutura do protocolo de um equipamento, ou seja, utilizando /h02 (byte STX 0x02) + "DeviceName" + /h03 (byte ETX 0x03).

O pacote **RX** é montado com o formato "(02" (inicializador não incluso STX 0x02) + "%s" + ")03" (terminador não incluso ETX 0x03), em que o parâmetro %s extrai os dados no formato **String**.

Exemplo de utilização

Protocolo	Objetivo	Configuração

TX: <STX>DeviceName<ETX> RX: <STX>Personal Computer <ETX>	Enviar o comando DeviceName a um equipamento e extrair uma <i>substring</i> do valor da resposta	• ID: 1 (um) • TX: /h02DeviceName/h03 • RX: (02%@r#as)03 • N1/B1: 0 (zero, envia e recebe comando) • N2/B2: 1 (um, ID do comando) • Read Tag: "son"
---	---	--

O pacote **TX** é montado para enviar o comando **DeviceName** conforme a estrutura do protocolo de um equipamento, ou seja, utilizando /h02 (byte STX 0x02) + "DeviceName" + /h03 (byte ETX 0x03).

O pacote **RX** é montado com o formato "(02" (inicializador não incluso STX 0x02) + "%@r#as" + ")03" (terminador não incluso ETX 0x03), em que o parâmetro %@r#as extrai uma *substring* entre o caractere inicial "r" e o caractere final "a" do valor "Personal Computer", resultando na *substring* "son".

Referência de Tags

Todos os Tags deste Driver devem ser referenciados pelos parâmetros *N/B*, de acordo com a tabela a seguir.

Opções disponíveis para os parâmetros de Tags

PARÂMETRO	DESCRIÇÃO
N1/B1	Código da função
N2/B2	Índice da listagem de comandos
N3/B3	Parâmetro adicional, se necessário
N4/B4	Parâmetro adicional, se necessário

A tabela a seguir contém as funções disponíveis neste Driver.

Funções disponíveis no Driver ASCII Generic

N1/B1	DESCRIÇÃO	APLICAÇÃO
0	Envia e Recebe Comandos	Utilizada para realizar uma transação completa com um equipamento, enviando um pacote de requisição TX e recebendo um pacote de resposta RX
1	Valores TX	Utilizada somente para definir as variáveis % do pacote TX , sem realizar comunicação com um equipamento

N1/B1	DESCRIÇÃO	APLICAÇÃO
2	Valores RX	Utilizada somente para consultar as variáveis % do pacote RX , sem realizar comunicação com um equipamento
3	Recebe Comandos	Realiza apenas a recepção de um pacote de resposta RX , geralmente utilizada com equipamentos que estão constantemente enviando dados sem a necessidade de receber um pacote de requisição TX
4	Envia Comandos	Realiza apenas o envio de um pacote de requisição TX , geralmente utilizada com equipamentos que não efetuam o retorno de um pacote de resposta RX
5	Envia o Valor do Tag	Utiliza o conteúdo de um Tag para representar o pacote de requisição TX e envia a um equipamento
6	Envia e Recebe Comandos de Lista	Utilizada para disparar um pacote de requisição TX que gera o retorno de múltiplos pacotes de resposta RX
7	Recebe Comandos de Lista	Realiza apenas a recepção de múltiplos pacotes de resposta RX

Envia e Recebe Comandos

Leitura e Escrita

N1	0 (zero)
N2	Índice de comando
N3	Índice de variável
N4	<i>Offset</i> da mensagem RX

Utilizado para realizar uma transação completa com um equipamento, enviando um pacote de requisição **TX** e recebendo um pacote de resposta **RX**.

Se utilizado em um comando de leitura, retorna os campos variáveis da mensagem de recepção, um em cada Elemento de Bloco, se aplicável, a partir do índice de variável informado no parâmetro *N3*. Por exemplo, se o comando **20** prevê uma resposta do tipo **ABC=%d** e **XYZ=%f** ao ler um Tag Bloco com dois Elementos na configuração **0.20.0.0**, o primeiro Elemento contém o primeiro campo variável (decimal) e o segundo Elemento contém o segundo campo variável (número real).

Use o parâmetro *N4* para que o comando analise o *frame* de resposta com um *offset* em relação ao primeiro caractere da mensagem. Por exemplo, para descartar os 100 primeiros bytes de uma mensagem, configure o parâmetro *N4* com o valor 100.

Valores TX

Leitura e Escrita

N1	1 (um)
N2	Índice de comando
N3	Índice de variável
N4	0 (zero)

Utilizado somente para definir as variáveis % do pacote **TX**, sem realizar comunicação com um equipamento. Este Tag lê ou escreve os valores dos campos variáveis % do pacote **TX** antes de enviar um comando. O tipo de dados de cada Tag deve coincidir com os tipos de dados informados na definição do comando. Se utilizado em um comando de leitura, são retornados os campos variáveis da mensagem de **TX**, um em cada Elemento de Bloco, se utilizado um Tag Bloco, a partir do índice de variável informado no parâmetro *N3*.

Valores RX

Somente Leitura

N1	2 (dois)
N2	Índice de comando
N3	Índice de variável
N4	0 (zero)

Utilizado somente para consultar as variáveis % do pacote **RX**, sem realizar comunicação com um equipamento. Este Tag lê os valores dos campos variáveis % do pacote **RX** atualizados na última recepção do comando informado no parâmetro *N2*. O tipo de dados de cada Tag coincide com os tipos de dados informados na definição do comando. Retorna os campos variáveis da mensagem de **RX**, um em cada Elemento de Bloco, se utilizado um Tag Bloco, a partir do índice de variável informado no parâmetro *N3*.

Recebe Comandos

Somente Leitura

N1	3 (três)
----	----------

N2	Índice de comando
N3	Índice de variável
N4	<i>Offset</i> da mensagem RX

Realiza apenas a recepção de um pacote de resposta **RX**, geralmente utilizado com equipamentos que enviam dados de forma contínua, sem a necessidade de transmitir um pacote de requisição **TX**. Retorna os campos variáveis da mensagem de recepção, um em cada Elemento de Bloco, se utilizado um Tag Bloco, a partir do índice de variável informado no parâmetro *N3*.

Use o parâmetro *N4* para que o comando analise o *frame* de resposta com um *offset* em relação ao primeiro caractere da mensagem. Por exemplo, para descartar os 100 primeiros bytes de uma mensagem, configure o parâmetro *N4* com o valor 100.

Para uma melhor recepção de dados contínuos, o usuário pode configurar o tempo de *time-out* do **IOKit** para um valor bem reduzido, algo em torno de 200 ms. Nesse tipo de recepção contínua de dados, o *time-out* não tem a função de monitorar um limite para as respostas de requisições **TX**, pelo simples fato de estes limites não estarem presentes neste tipo de comunicação.

Envia Comandos

Somente Escrita

N1	4 (quatro)
N2	Índice de comando
N3	0 (zero)
N4	Limpeza da resposta (opcional)

Realiza apenas o envio de um pacote de requisição **TX**, geralmente utilizado com equipamentos que não efetuam o retorno de um pacote de resposta **RX**. Indique no parâmetro *N4* um valor diferente de 0 (zero) se este Driver deve esperar uma resposta a este comando.

Envia o Valor do Tag

Somente Escrita

N1	5 (cinco)
N2	0 (zero)
N3	0 (zero)

N4	Limpeza da resposta (opcional)
----	--------------------------------

Utiliza o conteúdo do Tag para representar o pacote de requisição **TX** e envia a um equipamento. A mensagem a ser enviada é o conteúdo deste Tag em formato **Texto**, geralmente montado com funções em código de script de uma aplicação. Indique no parâmetro *N4* um valor diferente de 0 (zero) se este Driver deve esperar uma resposta a este comando.

Envia e Recebe Comandos de Lista

Somente Leitura

N1	6 (seis)
N2	Índice do comando
N3	Índice da variável
N4	Offset da mensagem RX

Utilizado para disparar um pacote de requisição **TX** que gera o retorno de múltiplos pacotes de resposta **RX**. Este Tag é similar ao comando com o parâmetro N1 igual a 0 (zero), porém gera uma lista de valores para cada conjunto repetido de variáveis encontradas, de acordo com a descrição do comando **RX**. Deve-se utilizar este Tag quando não se sabe quantas linhas o comando retorna, partindo do princípio que todas as linhas seguem uma mesma estrutura, conforme o exemplo a seguir.

```
=>HIS <ENTER>
FEEDER 1 Date: 08/10/99 Time: 08:40:16.740
STATION A
# DATE TIME EVENT LOCAT CURR FREQ GRP SHOT TARGETS
1 02/01/99 08:33:00.365 TRIG $$$$$$ 1 60.00 3 2
2 01/31/99 20:32:58.361 ER $$$$$$ 231 60.00 2 2
3 01/29/99 07:30:11.055 AG T 9.65 2279 60.00 3 2 INST 50
```

Recebe Comandos de Lista

Somente Leitura

N1	7 (sete)
N2	Índice do comando
N3	Índice da variável
N4	Offset da mensagem RX

Realiza apenas a recepção de múltiplos pacotes de resposta **RX**. Este Tag é similar ao Tag Recebe Comandos, exceto que retorna listas no formato do Tag Envia e Recebe Comandos de Lista.

Relação de Caracteres ASCII

Códigos de caracteres ASCII

CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX
(nul)	0	0	0	(sp)	32	40	32	@	64	100	64
(soh)	1	1	1	!	33	41	33	A	65	101	65
(stx)	2	2	2	"	34	42	34	B	66	102	66
(etx)	3	3	3	#	35	43	35	C	67	103	67
(eot)	4	4	4	\$	36	44	36	D	68	104	68
(enq)	5	5	5	%	37	45	37	E	69	105	69
(ack)	6	6	6	&	38	46	38	F	70	106	70
(bel)	7	7	7	'	39	47	39	G	71	107	71
(bs)	8	10	8	(	40	50	40	H	72	110	72
(ht)	9	11	9	)	41	51	41	I	73	111	73
(nl)	10	12	10	*	42	52	42	J	74	112	74
(vt)	11	13	11	+	43	53	43	K	75	113	75
(np)	12	14	12	,	44	54	44	L	76	114	76
(cr)	13	15	13	-	45	55	45	M	77	115	77
(so)	14	16	14	.	46	56	46	N	78	116	78
(si)	15	17	15	/	47	57	47	O	79	117	79

CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX
(dle)	16	20	16	0	48	60	48	P	80	120	80
(dc1)	17	21	17	1	49	61	49	Q	81	121	81
(dc2)	18	22	18	2	50	62	50	R	82	122	82
(dc3)	19	23	19	3	51	63	51	S	83	123	83
(dc4)	20	24	20	4	52	64	52	T	84	124	84
(nak)	21	25	21	5	53	65	53	U	85	125	85
(syn)	22	26	22	6	54	66	54	V	86	126	86
(etb)	23	27	23	7	55	67	55	W	87	127	87
(can)	24	30	24	8	56	70	56	X	88	130	88
(em)	25	31	25	9	57	71	57	Y	89	131	89
(sub)	26	32	26	:	58	72	58	Z	90	132	90
(esc)	27	33	27	;	59	73	59	[	91	133	91
(fs)	28	34	28	<	60	74	60	\	92	134	92
(gs)	29	35	29	=	61	75	61	]	93	135	93
(rs)	30	36	30	>	62	76	62	^	94	136	94
(us)	31	37	31	?	63	77	63	_	95	137	95

Documentação das Interfaces de Comunicação

Esta seção contém a documentação das Interfaces de Comunicação referentes ao Driver **ASCII**.

Esta documentação é compartilhada e mantida no manual do IOKit:

- Configurações de um Driver
- Configurações Gerais

- Configuração de Estatísticas
- Configuração da Interface Ethernet
- Configuração da Interface Modem
- Configuração da Interface RAS
- Configuração da Interface Serial

Histórico de Revisões do Driver

VERSÃO	DATA	AUTOR	COMENTÁRIOS
2.0.5	10/08/2026	M. Ludwig	• Driver atualizado para a biblioteca IOKit versão 3 e Visual Studio 2022 (<i>Case 37956</i>).
2.0.4	13/09/2023	C. Mello	• Ajustes para preservar o último dado válido recebido em leitura contínua de dados usando um Tag com o parâmetro <i>N1/B1</i> igual a 3 (três, Recebe Comandos) (<i>Case 34817</i>).
2.0.3	29/08/2019	C. Mello	• Atualização de plataforma no código fonte deste Driver (<i>Case 27445</i>).
2.0.2	22/05/2018	C. Mello	• Adicionado um parâmetro para descartar <i>n</i> bytes após encontrar o byte terminador de recepção de dados (<i>Case 24265</i>).
2.0.1	04/12/2017	C. Mello	• Ajustes com o símbolo @ para extrair dados no formato %s sem incluir o caractere de inicialização (<i>Case 15340</i>). • Adicionada uma função para determinar o byte inicializador e o byte terminador de recepção de dados (<i>Case 17078</i>). • Atualização e padronização deste Driver para o novo padrão da biblioteca IOKit versão 2 (<i>Case 23539</i>).
1.11.1	06/04/2011	C. Mello	• Adicionado suporte para interpretação dos valores decimais no formato BCD no tipo de dados %n (<i>Case 12162</i>). • Adicionado suporte para a plataforma WinCE (<i>Case 10917</i>).
1.10.1	11/05/2010	C. Mello	• Corrigido um problema na recepção dos dados quando a mensagem RX inicia com o valor "/hxx" (<i>Case 11377</i>).
1.9.1	16/09/2009	M. Ludwig	• Driver atualizado para a biblioteca IOKit versão 1.15 . • Corrigido um erro ao escrever Tags Bloco (<i>Case 10725</i>).
1.8.1	13/05/2009	C. Mello	• Ajustes no comando com o parâmetro <i>N1</i> igual a 7 (sete) para recepção dos dados em forma de lista (<i>Case 10472</i>).

VERSÃO	DATA	AUTOR	COMENTÁRIOS
1.7.1	05/06/2008	C. Mello	• Ajustes para ignorar o caractere delimitador como parte integrante dos dados (<i>Case 9566</i>). • Ajustes no comando com o parâmetro <i>N1</i> igual a 5 (cinco) para dispensar consistências do parâmetro <i>N2</i>.
1.6.1	01/06/2007	C. Mello	• Ajustes para prevenir a duplicação das propriedades de configuração deste Driver após reiniciar em tempo de execução (<i>Case 8031</i>).
1.5.1	23/01/2007	C. Mello	• Adicionado o formato /hxx para informar caracteres especiais nos campos TX e RX dos parâmetros extras deste Driver (<i>Case 7526</i>). • Ajustes no parâmetro <i>N4/B4</i> para habilitar a recepção de dados através dos comandos com o parâmetro <i>N1/B1</i> igual a 4 (quatro) e 5 (cinco).
1.4.1	22/12/2005	C. Mello	• Definido o tipo de dados %b. • Ajustes na caixa de diálogo de parâmetros Extras deste Driver. • Revisão geral de todo o código fonte.
		M. Salvador	• Adicionado suporte a listas de mensagens fora de ordem.
1.3.1	20/07/2005	M. Salvador	• Configurado o número máximo de bytes e Elementos de Bloco. • Definido o tipo de dados %t. • Definido o caractere @ em início de Strings, e também definidos os caracteres em hexadecimal (hXX), tanto para o início como para o fim de Strings. • Definidos novos comandos com o parâmetro <i>N1/B1</i> igual a 6 (seis) e 7 (sete). • Adicionado um <i>offset</i> no <i>frame</i> de resposta RX com o parâmetro <i>N4</i>. • Corrigido um erro em que o tipo de dados %f não interpretava sinais. • Adicionada uma lista de caracteres ASCII na documentação.
1.2.1	12/07/2005	C. Mello	• Definido o tipo de dados %d como um valor decimal com sinal. • Definido o tipo de dados %u como um valor decimal sem sinal.
		M. Salvador	• Implementado um comando apenas para recepção de dados com o parâmetro <i>N1/B1</i> igual a 3 (três). • Implementado um comando apenas para envio de dados com o parâmetro <i>N1/B1</i> igual a 4 (quatro). • Implementado um comando apenas para envio de dados do conteúdo do Tag no formato Texto com o parâmetro <i>N1/B1</i> igual a 5 (cinco).

VERSÃO	DATA	AUTOR	COMENTÁRIOS
1.1.1	14/09/2004	M. Salvador	• Adicionado suporte a valores hexadecimais, <i>/r</i> e <i>/n</i>.
1.0.1	08/09/2004	M. Salvador	• Versão original deste Driver.