Comunicação no sistema DomoBus
Application Programming Interface

V1.2

Ultima revisão: 19-Out-05

ÍNDICE

1. Introdução

Este documento define a API de comunicação do sistema DomoBus a ser usada ao alto nível, i.e., no contexto de PCs ou outros sistemas que ofereçam comunicação TCP/IP baseada em sockets.
Com o objectivo de simplificar as comunicações entre aplicações DomoBus, foi concebido um serviço de comunicação – ComServ – que oferece um mecanismo de comunicação entre aplicações, independentemente da sua localização física. Os aspectos essenciais do ComServ são descritos adiante e são justificadas as principais opções tomadas. Em seguida apresenta-se alguns aspectos relacionados com a rede DomoBus, nomeadamente quanto ao formato das mensagens que aí circulam. Por último apresenta-se a API de comunicações que as aplicações DomoBus devem seguir, sendo descrito detalhadamente as características da implementação efectuada desse objecto que deve ser incorporado nas aplicações do sistema DomoBus.

2. Arquitectura do serviço de comunicações ComServ

O ComServ é constituído por um processo que analisa e encaminha as mensagens entre aplicações DomoBus. Essas aplicações podem estar localizadas no mesmo PC (ou outra plataforma equivalente), em PCs distintos ou em controladores domóticos DomoBus de baixo nível, caso em que, tipicamente, a comunicação será feita através da porta série ou USB.
A comunicação pressupõe a existência de uma pilha de protocolos TCP/IP com uma interface por sockets. A comunicação processa-se usando datagramas UDP. Esta opção foi tomada por razões de eficiência e para permitir uma baixa latência na comunicação.
O ComServ possui um porto predefinido que é do conhecimento de todas as aplicações. Uma aplicação que queira enviar uma mensagem para outra aplicação limita-se a escrevê-la no porto do ComServ.
Cada PC (ou máquina equivalente) possui um ComServ que serve de mediador em todas as comunicações que envolvam aplicações localizadas nessa máquina.

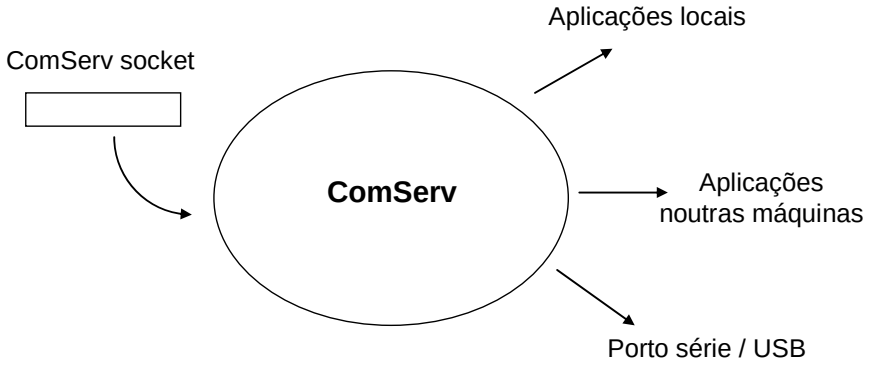

Figura 1 – Encaminhamento de mensagens por parte do ComServ

Cada aplicação possui um socket, criado quando a aplicação é iniciada, que é usado para receber mensagens.
Cada aplicação possui um endereço DomoBus (número de 32 bits). Esse endereço pode ser lido a partir de um ficheiro de configuração ou fornecido na linha de comandos quando a aplicação é iniciada.
Na fase de arranque a aplicação envia uma mensagem para o ComServ para se registar, indicando qual o seu endereço e qual o porto do seu socket de recepção de mensagens. Deste modo o ComServ toma conhecimento das aplicações existentes e de como pode comunicar com elas.

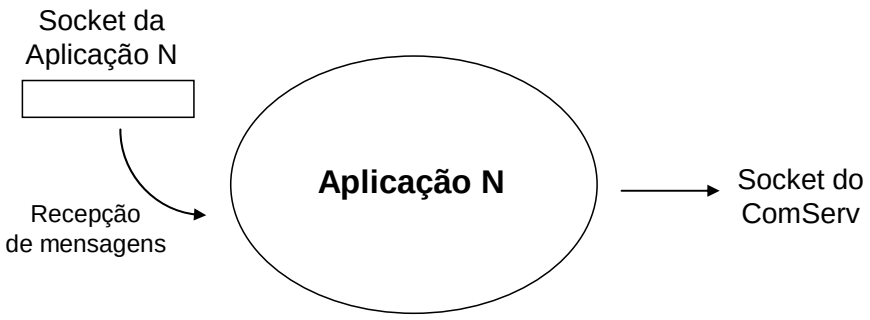

Figura 2 – Cada aplicação possui um socket para receber mensagens

Para facilitar o desenvolvimento das aplicações, pressupõe-se a existência de um objecto responsável pelas comunicações (ver figura 3). Esse objecto possui uma interface que constitui a presente API de comunicações. O objecto é responsável pela criação do socket, registo no ComServ e posterior recepção e envio de mensagens.
As mensagens DomoBus devem ser, na generalidade dos casos, confirmadas. Apenas deste modo existe garantia de que um dado comando enviado para uma aplicação foi recebido. Isto permite também contornar o problema de a comunicação UDP não ser fiável (mensagens podem ser perdidas sem que haja conhecimento desse facto).
Adicionalmente, para simplificar, assume-se um modelo de comunicação “at least once”, i.e., é possível saber se uma mensagem foi recebida mas não existe garantia de que tenha sido recebida mais do que uma vez (se a mensagem chegou mas a confirmação se perdeu, a mensagem é reenviada podendo pois ser recebida mais do que uma vez). Para que não existam problemas, a comunicação deverá ser idempotente. Os comandos devem ser absolutos (se executados mais do que uma vez não afectam o sistema) e nunca assumir uma alteração em função de um estado prévio. Por exemplo, deverão existir comandos para ligar / desligar uma lâmpada e não comandos para mudar o estado da lâmpada. Se forem recebidas duas mensagens seguidas indicando para ligar uma dada lâmpada, isso não tem impacto no sistema e a lâmpada será ligada. Pelo contrário, se fossem recebidas duas mensagens de mudança de estado, a lâmpada ficaria no estado inicial o que não era o pretendido.

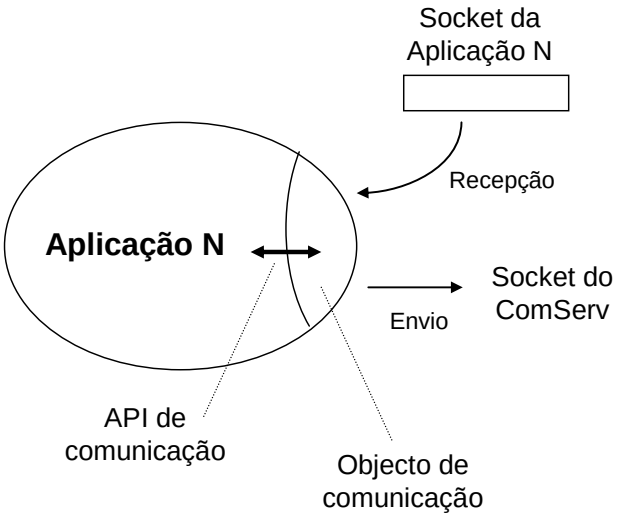

Figura 3 – As aplicações comunicam através de um objecto de comunicação

O objecto de comunicação lida com a retransmissão de mensagens. Se a resposta a uma mensagem não for recebida dentro de um certo intervalo de tempo, a mensagem será reenviada. Este processo pode repetir-se um certo número de vezes. Após esse número de tentativas, se não houve sucesso, a aplicação será informada desse facto.

3. A rede DomoBus

A rede DomoBus está dividida em segmentos que se ligam a controladores domóticos – nós. Cada nó tem associado um conjunto de aplicações domóticas e, cada uma destas, um conjunto de dispositivos domóticos (sensores e actuadores).

3.1 Endereçamento

Os dispositivos são endereçados univocamente por 32 bits, embora para o encaminhamento das mensagens apenas sejam usados os 24 bits mais significativos para identificar a aplicação.
Após a aplicação receber uma mensagem são analisados os 8 bits correspondentes ao endereço do dispositivo de modo a saber qual dos seus dispositivos está a ser referido.
Um dispositivo é identificado univocamente por 32 bits constituído pelos campos que se podem observar na figura 4.

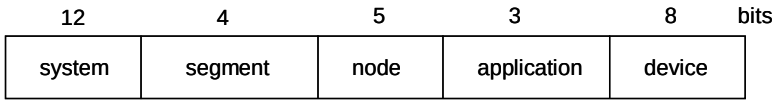

Figura 4 – Endereço de um dispositivo domótico na rede DomoBus.

As aplicações de alto nível a funcionar em PC não necessitam de tratar independentemente os campos do endereço, pois o ComServ fornece transparentemente o transporte e encaminhamento das mensagens entre estas aplicações e as aplicações da rede DomoBus. Assim sendo, as aplicações de alto nível, internamente, guardam os 32 bits do endereço. Para enviar dados entre si (comunicação entre aplicações) têm de utilizar as mensagens DATA do ComServ, explicadas mais à frente. Estas mensagens são endereçadas a outras aplicações e consequentemente apenas são relevantes os 24 bits iniciais do endereço referentes à aplicação. Os restantes 8 bits referentes ao dispositivo devem estar a zero. Se a mensagem for endereçada a uma aplicação do DomoBus a informação com a identificação do dispositivo irá no campo de dados uma vez que o cabeçalho das mensagens do DomoBus apenas contém informação dos endereços das aplicações.

3.2 Mensagens entre Aplicações e a rede DomoBus

Como foi visto anteriormente, as mensagens trocadas entre uma aplicação e a rede DomoBus têm de passar inicialmente pelo ComServ que faz o seu encaminhamento. Assim, as aplicações devem lidar o formato das mensagens DATA do ComServ para comunicar com outras aplicações. As funções do objecto de comunicação permitem manipular facilmente este tipo de mensagens, particularmente as mensagens standard.
Na figura 5 está representado todo o formato das mensagens que interessam para a presente especificação.

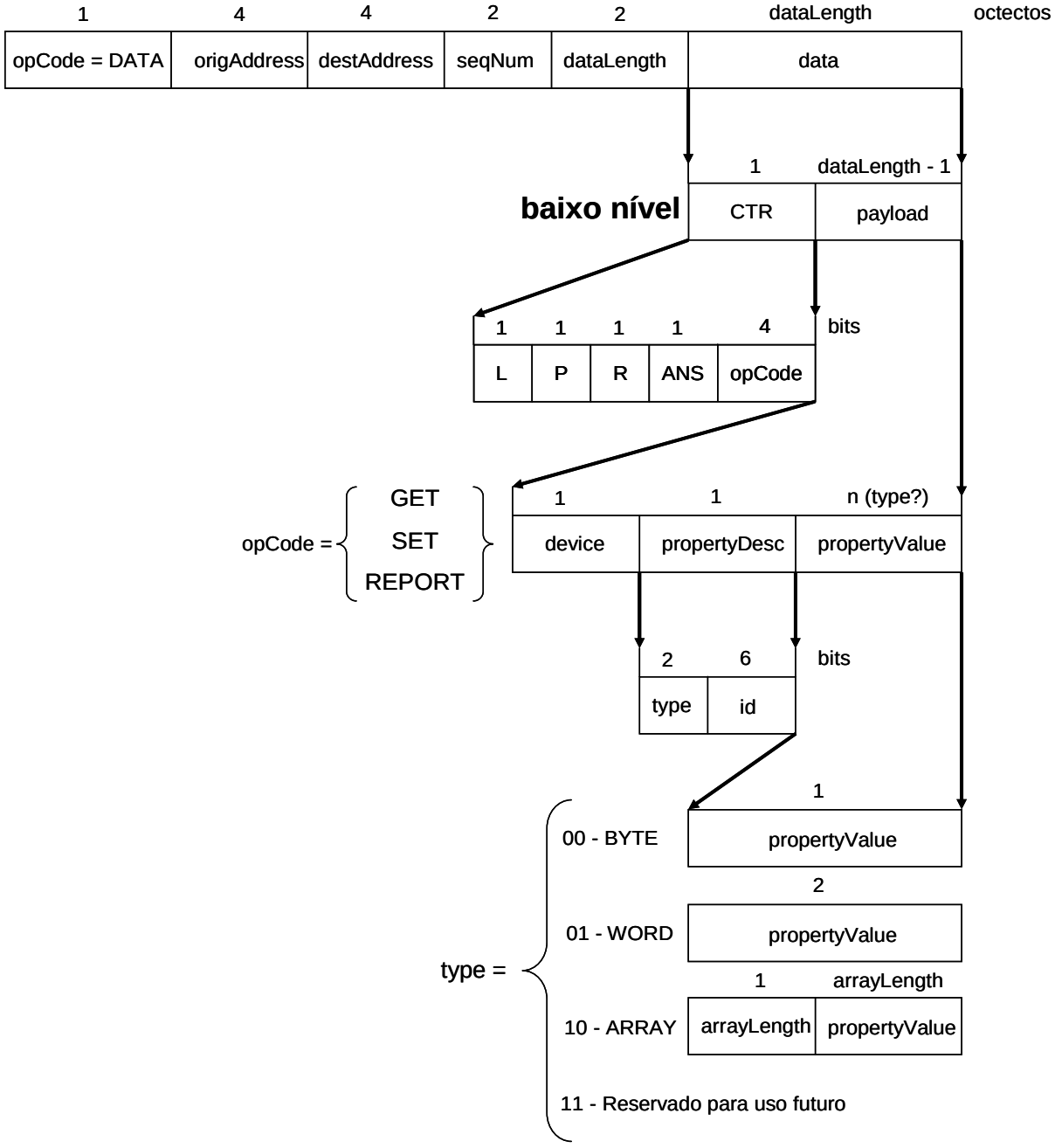

Figura 5 – Formatos das mensagens entre aplicações e a rede DomoBus.

4. Objecto de comunicação – overview

Nesta secção apresenta-se uma descrição geral do objecto de comunicação e do seu funcionamento.
O objecto de comunicação foi implementado na linguagem C, para ambiente Windows e utiliza a versão 2 do Windows Sockets (biblioteca *WSOCK32.LIB*). Este objecto materializa-se na biblioteca dinâmica **DomoBus_COMM.DLL** a qual pode ser usada directamente ao se desenvolver nas linguagens C e C++, necessitando apenas de fazer a importação dos cabeçalhos da referida API disponíveis na *DomoBus_COMM.LIB*.
Está também disponível uma adaptação da API descrita neste documento para a linguagem C#. De notar que para essa linguagem de programação, embora se utilize a mesma DLL de base (aqui explicada), a API em si é um pouco diferente devido ao próprio conceito de orientação por objectos subjacente a essa linguagem. No entanto, dispõe de todas as funcionalidades aqui descritas. Para mais informações consultar respectiva documentação.

A implementação efectuada tem em conta a possibilidade de as aplicações poderem ter várias threads e elas poderem querer enviar mensagens em concorrência umas com as outras. Já no caso da recepção isso não é verdade, i.e., uma aplicação não poderá receber mensagens simultaneamente mas apenas numa única thread. Caso a aplicação tenha threads concorrentes a chamar a função DomoBus_GetMsg(), que permitem obter mensagens, acontece que, quando chegar uma nova mensagem esta será apenas entregue à thread que tenha sido a última a executar uma dessas funções.

O objecto de comunicação é único e possui duas threads. Uma de leitura do socket, a qual está bloqueada enquanto não chegar uma mensagem, e outra que trata do reenvio de mensagens que ainda não tenham obtido resposta. Para tal, o objecto dispõe de um conjunto de semáforos. Quando recebe um pedido de envio, a mensagem é escrita no socket do ComServ e é guardada uma cópia numa estrutura de dados para o caso de ser necessário o seu reenvio.

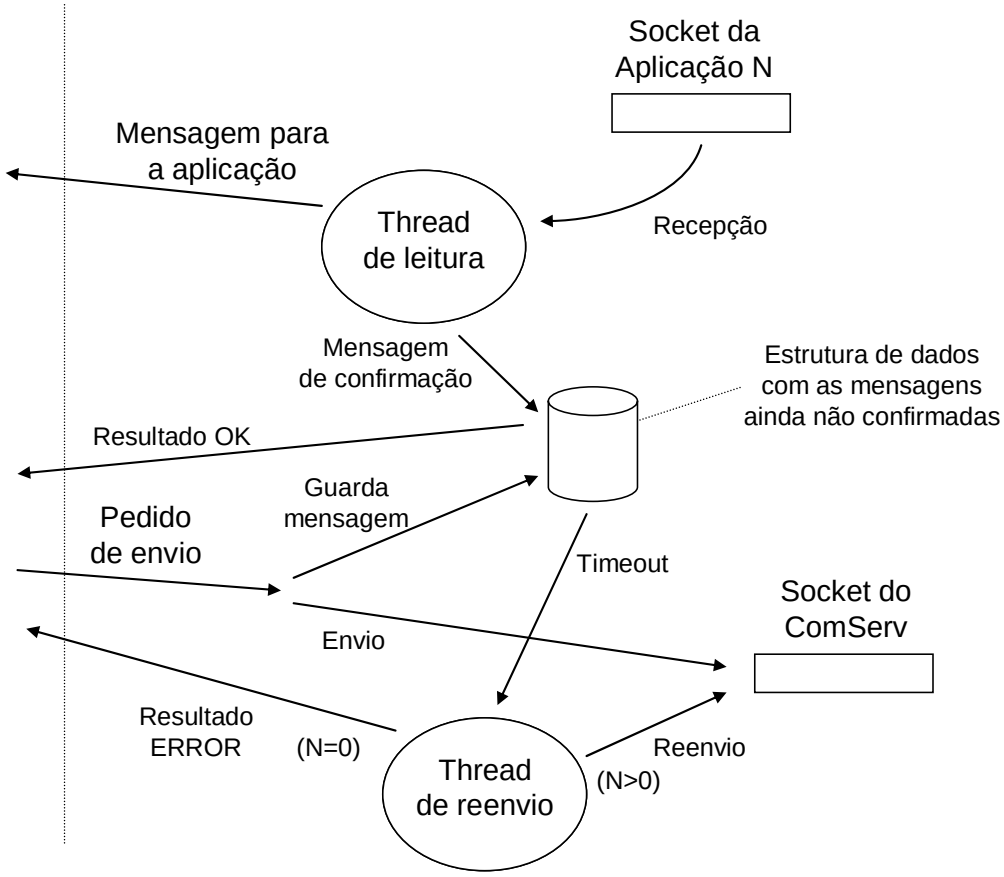

Figura 4 – Arquitectura interna do objecto de comunicação

Quando chega uma nova mensagem ela é analisada para verificar se é uma resposta. Se não for, a mensagem é entregue à aplicação. Se for uma resposta, é analisada a estrutura de dados que contém as mensagens a aguardar resposta para desbloquear as funções que esperavam pela resposta. Se não estiver na estrutura de dados a mensagem é ignorada.
O conjunto de funções disponibilizadas pelo objecto de comunicação constitui a referida API de comunicações que é o assunto do presente documento. Esse leque de funções foi pensado para oferecer o máximo de funcionalidades a diferentes e variadas aplicações do sistema DomoBus. No entanto, poderão ser posteriormente estendidas numa nova versão do presente objecto de comunicação, consoante as necessidades específicas das aplicações que se venham a desenvolver.

5. API de comunicação DomoBus_COMM LIBRARY

Descreve-se em seguida os métodos e funcionalidades oferecidas pelo objecto de comunicação descrito no ponto anterior (DomoBus_COMM.DLL).

5.1 DomoBus_GetCommConfig()

Descrição:

Permite obter os valores actuais de vários parâmetros relacionados com as comunicações. Veja também **DomoBus_SetCommConfig()**.

#include "comm.h"

int DomoBus_GetCommConfig(

OUT	**int ***	*receiveMode,*
OUT	**int ***	*receiveTimeoutValue,*
OUT	**int ***	*sendTimeoutValue,*
OUT	**int ***	*retransmitNum,*
OUT	**int ***	*retransmitInterval*

);

receiveMode	Modo de recepção das mensagens que pode ser: DomoBus_BLOCK_NO_TIMEOUT, DomoBus_BLOCK_WITH_TIMEOUT, DomoBus_DO_NOT_BLOCK. Indica se **DomoBusGetMsg()** se bloqueia sem timeout (se não chegar mensagem não sai, fica sempre à espera; ignora *receiveTimeoutValue*), se bloqueia com timeout (*receiveTimeoutValue*) ou não se bloqueia (se tiver mensagem devolve-a, se não tiver devolve código indicando isso). Por omissão o modo de recepção é: DomoBus_BLOCK_WITH_TIMEOUT.
receiveTimeoutValue	Valor do tempo máximo (expresso em milisegundos) a esperar por uma mensagem na recepção, i.e., quando se chama a função **DomoBus_GetMsg()**. Por omissão este valor é de 10 segundos.
sendTimeoutValue	Valor do tempo máximo (expresso em milisegundos) a esperar pela resposta de confirmação a uma mensagem no envio, i.e., quando se chamam as funções **DomoBus_SendMsg** ou **DomoBus_SendMsgStd**. Por omissão este valor é de 3 segundos.

retransmitNum	Número de vezes que as mensagens devem ser retransmitidas se não for recebida uma resposta ao fim de um certo intervalo de tempo (*retransmitInterval*). Por omissão o número de retransmissões é de 3.
retransmitInterval	Valor do intervalo de tempo (expresso em milisegundos) a esperar para reenviar a próxima mensagem presente no buffer de reenvio, no caso de ainda não ter sido recebida uma resposta. Este valor depende das características da rede, por omissão é de 50 milisegundos, que deve ser afinado depois na prática.

Devolve:

Se nenhum erro ocorrer **DomoBus_GetCommConfig()** retorna sucesso – DomoBus_OK.

5.2 DomoBus_SetCommConfig()

Descrição:

Define os valores de vários relacionados com as comunicações.
Veja também **DomoBus_GetCommConfig()**.

#include "comm.h"

int DomoBus_SetCommConfig(

IN	**int**	*receiveMode,*
IN	**int**	*receiveTimeoutValue,*
IN	**int**	*sendTimeoutValue,*
IN	**int**	*retransmitNum,*
IN	**int**	*retransmitInterval*

);

Ver explicação dos parâmetros em **DomoBus_GetCommConfig()**.

Devolve:

Se nenhum erro ocorrer **DomoBus_SetCommConfig()** retorna sucesso – DomoBus_OK.

Notas:

Esta função pode ser chamada múltiplas vezes, definindo, a partir desse momento, o comportamento do objecto de comunicação.

De notar que esta função deve ser chamada antes de **DomoBus_CommOpen()**, no caso de se pretender utilizar um valor diferente dos valores por omissão para os parâmetros *retransmitNum* e *retransmitInterval*, os quais também são usados aquando do registo no ComServ.

No caso de se pretender que qualquer um dos parâmetros não seja alterado deve ser especificado NULL ou ZERO como valor desse parâmetro, excepto o parâmetro *retransmitNum* que, como pode ser zero, tem de ser sempre indicado.

5.3 DomoBus_CommOpen()

Descrição:

Efectua várias inicializações necessárias ao bom funcionamento do objecto de comunicação.
Cria o socket da aplicação e regista a aplicação no ComServ (ver secção 2). Fica a aguardar a confirmação do registo no ComServ até ocorrer timeout (erro).

#include "comm.h"

int DomoBus_CommOpen(

	IN	**DomoBus_ADDRESS**	*myAddress***,**
	IN	**DomoBus_COMMHANDLE**	*ComServPort***,**
	IN	**DomoBus_COMMHANDLE**	*myPort*

);

myAddress	Endereço da aplicação DomoBus (número de 32 bits). Esta informação é obtida, tipicamente, a partir de um ficheiro de configuração da aplicação.
ComServPort	Número do porto interno do ComServ. Se for indicado 0 ou NULL, será usado o porto interno do ComServ que por omissão é 65534.
myPort	Número do porto a ser criado pelo objecto de comunicação e que servirá para receber as mensagens destinadas à aplicação (se o porto indicado não estiver livre, o método devolve erro). Se for indicado 0 ou NULL, o objecto de comunicação usa um porto que esteja livre entre 1024 e 5000.

Devolve:

Se nenhum erro ocorrer **DomoBus_CommOpen()** retorna sucesso - DomoBus_OK. Caso contrário é retornado um dos seguintes erros:
DomoBus_COMMHANDLE_ERROR – erro na criação do socket da aplicação. Este erro também ocorre quando o *myPort* especificado não estiver disponível.
DomoBus_COMM_ERROR – erro na comunicação com o ComServ.
DomoBus_ERROR – outro tipo de erro.

Notas:

O tempo de espera pela confirmação de registo no ComServ, timeout ao fim do qual se retorna DomoBus_COMM_ERROR, corresponde ao produto dos parâmetros *retransmitInterval* por *retransmitNum* (ver **DomoBus_GetCommConfig()**). É pouco provável que necessite alterar os valores por omissão destes parâmetros para se conseguir registar no ComServ. Por isso, se acontecer este erro, verifique se o ComServ está a correr na mesma máquina onde está a executar a aplicação, se o *ComServPort* está correcto e se *myAddress* já se encontra configurado no ComServ.

DomoBus_CommOpen() pode ser chamado mais do que uma vez, podendo cada aplicação registar-se com 1, 2 ou mais endereços até um valor máximo de MAX_NUM_ADDR_BY_APP 50 (vários registos com endereços diferentes). Chama-se a função e indica-se um *myAddress*, volta-se a chamar a função e indica-se outro endereço diferente mas os portos a serem usados serão os mesmos, pelo que numa chamada a esta função que não seja a primeira vez apenas o parâmetro *myAddress* é relevante. Esta funcionalidade permite que uma mesma aplicação possa ter múltiplos endereços, o que em certas situações pode ser útil.

5.4 DomoBus_GetMsg()

Descrição:

Permite obter uma mensagem. O modo recepção dessa mensagem é definido por *receiveMode*, ver **DomoBus_GetCommConfig()**.

#include "comm.h"

int DomoBus_GetMsg(

	IN OUT	**int ***	*dataLength,*
	OUT	**DomoBus_DATA_PTR**	*data,*
	IN	**DomoBus_ADDRESS ***	*originAddress,*
	IN	**DomoBus_ADDRESS ***	*destinationAddress*

);

dataLength	Ponteiro para inteiro. Inicialmente contém o número máximo de bytes que podem ser recebidos, i.e., copiados para *data* (indica a dimensão de *data*). Neste argumento é devolvido o número de bytes escritos em *data*. De notar que o tamanho máximo de uma mensagem é SOCK_BUFF_SIZE 500.
data	Ponteiro/referência para mensagem (array de bytes). A mensagem recebida é copiada para *data*.
originAddress	Endereço de origem da mensagem, i.e., o endereço da aplicação que envia a mensagem.
destinationAddress	Endereço de destino da mensagem, i.e., o endereço da aplicação ou dispositivo a quem se destina a presente mensagem.

Devolve:

Se nenhum erro ocorrer **DomoBus_GetMsg()** retorna sucesso - DomoBus_OK ou DomoBus_NO no caso do modo de recepção ser DomoBus_DO_NOT_BLOCK e não exista nenhuma mensagem para receber.
Caso contrário é retornado um dos seguintes erros:
DomoBus_CALLBACK_ACTIVE – o modo de recepção por "callback" está activo, i.e., já foi chamada a função **DomoBus_DefineReceiveFunction()**. Para desactivar este modo utilize **DomoBus_DefineReceiveFunction(** NULL, 0 **)**.
DomoBus_TIMEOUT – passou o tempo especificado em *receiveTimeoutValue* (ver **DomoBus_GetCommConfig()**) e não chegou nenhuma mensagem.
DomoBus_MSGSIZE_ERROR – quando o tamanho da mensagem recebida for maior que o tamanho especificado em *dataLength*. De notar que neste caso a mensagem é perdida.
DomoBus_ERROR – outro tipo de erro, nomeadamente um SOCKET_ERROR.

Notas:

IMPORTANTE: Uma vez estando a utilizar o modo de recepção DomoBus_BLOCK_NO_TIMEOUT a função é bloqueante, pelo que a sua evocação deverá ser feita a partir de uma thread específica de leitura de mensagens, caso se pretenda que a aplicação possa realizar outras operações enquanto aguarda a chegada de uma mensagem. Em alternativa poderá usar-se o modo DomoBus_BLOCK_WITH_TIMEOUT e assim, pode jogar-se com o parâmetro *receiveTimeoutValue*.
O modo de recepção DomoBus_DO_NOT_BLOCK serve para verificar se existe alguma mensagem disponível para a aplicação e caso exista permite obtê-la imediatamente.

Veja também **DomoBus_DefineReceiveFunction()**.

5.5 DomoBus_ParseStdMsg()

Descrição:

Permite facilitar a leitura de mensagens em formato standard, fazendo o respectivo parsing.

#include "comm.h"

int DomoBus_ParseStdMsg(

IN	**int**	*dataLength,*
IN	**DomoBus_DATA_PTR**	*data,*
OUT	**DomoBus_CTR ***	*controlValue,*
OUT	**DomoBus_PROPERTY_DESC ***	*propertyDesc,*
OUT	**DomoBus_VALUE_PTR**	*propertyValue,*
OUT	**DomoBus_ADDRESS ***	*destinationAddress*

);

dataLength	Tamanho (número de bytes) da mensagem *data*.
data	Ponteiro/referência para mensagem da qual se pretende fazer o parsing no caso se ser uma mensagem standard (GET, SET, REPORT).
controlValue	Byte de controlo.
propertyDesc	Descritor de propriedade.
propertyValue	Valor de propriedade.

destinationAddress Endereço de destino da mensagem, que no caso de se estar a fazer o parsing de uma mensagem recebida será o próprio endereço da aplicação.

Devolve:

Caso a mensagem passada em *data* seja uma mensagem standard (GET, SET, REPORT) esta função retorna DomoBus_OK. Senão retorna DomoBus_ERROR.

Notas:

Deve ser sempre testado por parte da aplicação se a função retornou DomoBus_OK, pois só assim se tem a garantida que a mensagem era standard e portanto os valores dos parâmetros devolvidos serão os correctos.
Ver **DomoBus_SendStdMsg()** para esclarecimentos adicionais sobre o byte de controlo e o descritor de propriedade.

5.6 DomoBus_DefineReceiveFunction()

Descrição:

Define a função que deverá ser evocada para tratar uma mensagem de chegada (mecanismo de "callback").

#include "comm.h"

int DomoBus_DefineReceiveFunction (
IN **DomoBus_FUNCTION_PTR** *receiveFunction*
IN **int** *dataMaxLength*
);

receiveFunction Ponteiro/referência para uma função “callback”.

dataMaxLength Indica número máximo de bytes da mensagem que pode ser recebida pela respectiva função.

Devolve:

Se nenhum erro ocorrer **DomoBus_DefineReceiveFunction()** retorna sucesso – DomoBus_OK.
Caso contrário é retornado DomoBus_ERROR.

Notas:

A função *receiveFuncion* deve ter quatro parâmetros que são os mesmos de **DomoBus_GetMsg()**. A única diferença é que o parâmetro *dataLength* inicialmente não deverá conter o número máximo de bytes que podem ser recebidos, uma vez que isto é o papel do parâmetro *dataMaxLength* da presente função. Um exemplo do cabeçalho desta função é:

*void CALLBACK receiveFuncion(int *dataLength,*
DomoBus_DATA_PTR data,
*DomoBus_ADDRESS *originAddress,*
*DomoBus_ADDRESS *destinationAddress)*

IMPORTANTE: A função indicada, que processará cada mensagem recebida, não pode ser bloqueante. Essa função deverá analisar a mensagem recebida e tratá-la de imediato actualizando, por exemplo, estruturas de dados. Isto deve-se a que a função irá correr do contexto de uma thread do objecto de comunicação pelo que, em caso de bloqueio, isso impediria a recepção de novas mensagens. No caso de o tratamento das mensagens obrigar a acções que possam ser bloqueantes, em vez desta técnica de "callback" deverá ser usado antes o método **DomoBus_GetMsg()**.

Depois de ser usada esta função não é permitido chamar a função **DomoBus_GetMsg()**, mas o contrário é permitido, i.e., até determinada altura pode usar-se o **DomoBus_GetMsg()**, e depois pode passar a usar-se o **DomoBus_DefineReceiveFunction()** para obter as mensagens que chegam para a aplicação. No entanto, é possível desactivar este mecanismo de "callback" e passar a usar **DomoBus_GetMsg()**, para tal deve voltar a chamar-se **DomoBus_ DefineReceiveFunction(** NULL, 0 **)**. Estas duas funções foram concebidas para serem usadas alternativamente.

5.7 DomoBus_SendMsg()

Descrição:

Envia a mensagem indicada. Esta função é bloqueante. Só retorna quando for recebida a resposta à mensagem enviada ou em caso de erro de espera, i.e., foi desbloqueada por um timeout.
Pode ser especificado o tempo máximo que se está disposto a esperar por uma mensagem, independentemente dos parâmetros de retransmissão definidos (ver s*endTimeoutValue* em **DomoBus_GetCommConfig()**), sendo que esse valor deve ser suficiente para que se efectuem as *retransmitNum* retransmissões.

#include "comm.h"

int DomoBus_SendMsg(

IN	**int**	*dataLength,*
IN OUT	**DomoBus_DATA_PTR**	*data,*
IN	**DomoBus_ADDRESS**	*originAddress,*
IN	**DomoBus_ADDRESS**	*destinationAddress*

);

dataLength Número de bytes da mensagem a enviar.

data Ponteiro/referência para a mensagem a enviar (array de bytes). Aqui também é devolvida a mensagem de resposta.

originAddress Endereço de origem da mensagem, i.e., o endereço da aplicação que envia a mensagem.

destinationAddress Endereço de destino da mensagem, i.e., o endereço da aplicação ou dispositivo a quem se destina a presente mensagem.

Devolve:

Se nenhum erro ocorrer **DomoBus_SendMsg()** retorna sucesso – DomoBus_OK. Caso contrário é retornado um dos seguintes erros:
DomoBus_TIMEOUT – passou o tempo especificado em *sendTimeoutValue* (ver **DomoBus_GetCommConfig()**) e não chegou nenhuma resposta à mensagem enviada.
DomoBus_ADDRESS_ERROR – o endereço que especificou como sendo o endereço de origem não está registado. Para usar tal endereço como sendo mais um endereço da aplicação terá de fazer **DomoBus_CommOpen().**
DomoBus_ERROR – Outro tipo de erro, nomeadamente um SOCKET_ERROR.

5.8 DomoBus_SendStdMsg()

Descrição:

Constrói uma mensagem a partir dos argumentos e envia-a.
(Veja também **DomoBus_SendMsg()**).

#include "comm.h"

int DomoBus_SendStdMsg (

IN	**DomoBus_CTR**	*controlValue,*
IN	**DomoBus_PROPERTY_DESC**	*propertyDesc,*
IN OUT	**DomoBus_VALUE_PTR**	*propertyValue*
IN	**DomoBus_ADDRESS**	*originAddress,*
IN	**DomoBus_ADDRESS**	*destinationAddress*

);

controlValue	Byte de controlo que inclui o código da operação normalizada a realizar (GET, SET, REPORT).
propertyDesc	Descritor de propriedade (tipo de propriedade e identificador da propriedade - número)
propertyValue	Ponteiro para valor de propriedade. No caso de ser uma mensagem GET neste parâmetro é devolvida a respectiva resposta.
originAddress	Endereço de origem da mensagem, i.e., o endereço da aplicação que envia a mensagem.
destinationAddress	Endereço de destino da mensagem, i.e., o endereço da aplicação ou dispositivo a quem se destina a presente mensagem.

Devolve:

Se nenhum erro ocorrer **DomoBus_SendStdMsg()** retorna sucesso – Caso contrário é retornado um dos seguintes erros:
DomoBus_TIMEOUT – passou o tempo especificado em *sendTimeoutValue* (ver **DomoBus_GetCommConfig()**) e não chegou nenhuma resposta à mensagem enviada.
DomoBus_ADDRESS_ERROR – o endereço que especificou como sendo o endereço de origem não está registado. Para usar tal endereço como sendo mais um endereço da aplicação terá de fazer **DomoBus_CommOpen().**
DomoBus_ERROR – Outro tipo de erro, nomeadamente um SOCKET_ERROR.

5.9 DomoBus_SendMsgAsynch()

Descrição:

Este método limita-se a escrever a mensagem directamente no socket de envio e retorna (não é bloqueante).

#include "comm.h"

int DomoBus_SendMsgAsynch(

IN	**int**	*dataLength,*
IN	**DomoBus_DATA_PTR**	*data,*
IN	**DomoBus_ADDRESS**	*originAddress,*
IN	**DomoBus_ADDRESS**	*destinationAddress*

);

dataLength	Número de bytes da mensagem.
data	Ponteiro/referência para a mensagem a enviar (array de bytes).
originAddress	Endereço de origem da mensagem, i.e., o endereço da aplicação que envia a mensagem.
destinationAddress	Endereço de destino da mensagem, i.e., o endereço da aplicação ou dispositivo a quem se destina a presente mensagem.

Devolve:

Se nenhum erro ocorrer **DomoBus_SendMsgAsynch()** retorna sucesso – DomoBus_OK . Caso contrário é retornado um dos seguintes erros:
DomoBus_ADDRESS_ERROR – o endereço que especificou como sendo o endereço de origem não está registado. Para usar tal endereço como sendo mais um endereço da aplicação terá de fazer **DomoBus_CommOpen().**
DomoBus_ERROR – Outro tipo de erro, nomeadamente um SOCKET_ERROR.

5.10 DomoBus_SendAnswer()

Descrição:

Esta função serve para enviar uma mensagem de resposta a uma outra mensagem recebida.

#include "comm.h"

int DomoBus_SendAnswer(

IN	**int**	*dataLength,*
IN	**DomoBus_DATA_PTR**	*data,*
IN	**DomoBus_ADDRESS**	*originAddress,*
IN	**DomoBus_ADDRESS**	*destinationAddress*

);

dataLength	Número de bytes da mensagem .
data	Ponteiro/referência para a mensagem a enviar (array de bytes).
originAddress	Endereço de origem da mensagem, i.e., o endereço da aplicação que envia a mensagem.
destinationAddress	Endereço de destino da mensagem, i.e., o endereço da aplicação ou dispositivo a quem se destina a presente mensagem.

Devolve:

Se nenhum erro ocorrer **DomoBus_ SendAnswer()** retorna sucesso – DomoBus_OK.
Caso contrário é retornado DomoBus_ERROR.

Notas:

Os parâmetros desta função referem-se à mensagem recebida, que se assume que já foi processada. Esta função troca o endereço de destino com o endereço de origem e activa o bit ANS (answer) do campo de controlo e chama a função não bloqueante **DomoBus_SendMsgAsynch()** com uma diferença, no caso de **DomoBus_SendAnswer()** a função **DomoBus_SendMsgAsynch()** não guarda a mensagem na estrutura de dados para que se obtenha uma confirmação de envio, i.e., não há respostas a respostas.

Se a mensagem recebida foi um GET, na posição adequada de *data* deverá ser colocado o valor da propriedade solicitada antes de chamar este método.

5.11 DomoBus_CommClose()

Descrição:

Esta função deve ser chamada quando não se pretende usar mais o objecto de comunicação, tipicamente quando a aplicação vai terminar execução.
Comunica com o ComServ e cancela os registos anteriormente efectuados.

#include "comm.h"

int DomoBus_CommClose(void) ;

Devolve:

Se nenhum erro ocorrer **DomoBus_CommClose()** retorna sucesso – DomoBus_OK.
Caso contrário é retornado DomoBus_ERROR.

Notas:

Esta função repõe o estado inicial do objecto de comunicação pelo que posteriormente a aplicação pode voltar a chamar **DomoBus_CommOpen()**.

5.12 DomoBus_SetCommVerbose()

Descrição:

Esta função activa a execução do objecto de comunicação em modo verboso. Esta função é útil essencialmente para *debug*, pois imprime mensagens de estado e também todos os tipos de erro que vão ocorrendo.

#include "comm.h"

DomoBus_SetCommVerbose(
int *verboseMode*
);

verboseMode	Modo que pode ser: DomoBus_COMM_VERBOSE_TO_CONSOLE DomoBus_COMM_VERBOSE_TO_FILE .

Devolve:

Se nenhum erro ocorrer **DomoBus_SetCommVerbose()** retorna sucesso – DomoBus_OK.
Caso contrário é retornado DomoBus_ERROR.

Anexo A – Ficheiro *"comm.h"*

```
#ifndef _DOMOBUS_COMM_API_H_
#define _DOMOBUS_COMM_API_H_

/*-------------------------------------------------------------------------*/
#define IMPORT_A_DLL
//#define MAKE_A_DLL
/*-------------------------------------------------------------------------*/
#ifdef MAKE_A_DLL
#define LINKDLL __declspec( dllexport )
#ifdef IMPORT_A_DLL
#define LINKDLL __declspec( dllimport )
#endif
#else
#define LINKDLL
#endif

/*-------------------------------------------------------------------------*/
/* definicao de tipos básicos, que também estão contidos no windef.h */
typedef int                 BOOL;
typedef unsigned char       BYTE;
typedef unsigned short      WORD; /*2 BYTES*/
typedef unsigned long       DWORD;/*4 BYTES*/

#ifndef FALSE
#define FALSE               0
#endif

#ifndef TRUE
#define TRUE                1
#endif

#define CALLBACK    __stdcall
/*-------------------------------------------------------------------------*/

/* Modo verboso - serve para imprimir mensagens de erro e de estado no ecrá ou
num ficheiro */
#define DomoBus_COMM_VERBOSE_TO_CONSOLE             1
#define DomoBus_COMM_VERBOSE_TO_FILE                2

/* Formato DomoBus */

#define DomoBus_YES             1
#define DomoBus_NO              0

#define DomoBus_OK              1

/*MODO DE RECEPÇÃO DE MENSAGENS:*/
#define DomoBus_BLOCK_NO_TIMEOUT 1
#define DomoBus_BLOCK_WITH_TIMEOUT      2
#define DomoBus_DO_NOT_BLOCK            3

/*MENSAGENS DE ERRO:*/
#define DomoBus_ERROR                   (-1)
#define DomoBus_COMM_ERROR              (-2)
#define DomoBus_COMMHANDLE_ERROR        (-3)
#define DomoBus_MSGSIZE_ERROR           (-4)
#define DomoBus_CALLBACK_ACTIVE         (-5)
#define DomoBus_ADDRESS_ERROR           (-6)
```

```
#define DomoBus_TIMEOUT                         (-10)


typedef DWORD DomoBus_ADDRESS;

typedef WORD DomoBus_COMMHANDLE;

typedef BYTE* DomoBus_VALUE_PTR;

typedef BYTE* DomoBus_DATA_PTR;

typedef void(CALLBACK *DomoBus_FUNCTION_PTR)(int*, DomoBus_DATA_PTR,
DomoBus_ADDRESS*, DomoBus_ADDRESS*);


/* estrutura do Byte de Controlo da trama DomoBus */
typedef struct DomoBus_CTR
{
      unsigned opCode : 4;      /*4 bits de opCode*/
      unsigned ANS : 1;         /*Answer - resposta*/
      unsigned R : 1;           /*bit de retransmissão*/
      unsigned P : 1;           /*bit de prioridade*/
      unsigned L : 1;           /*reservado para uso futuro*/
}DomoBus_CTR;

/* estrutura do descritor de propriedade PropertyDesc da trama DomoBus: */
typedef struct DomoBus_PROPERTY_DESC
{
      unsigned id   : 6; /* 6 bits do identificador da propriedade */
      unsigned type : 2; /* 2 bits do tipo da propriedade */
}DomoBus_PROPERTY_DESC;
/* tipo da propriedade: */
typedef enum
{
            PROPERTY_DESC_TYPE_BYTE,   /*0x00*/
            PROPERTY_DESC_TYPE_WORD,   /*0x01*/
            PROPERTY_DESC_TYPE_ARRAY,  /*0x10*/
      /*Reservado para uso futuro*/    /*0x10*/
}DomoBus_PROPERTY_DESC_TYPE;

/* Tipo de Mensagens: */
typedef enum
{
      MSG_TYPE_GET                    = 0x00,
      MSG_TYPE_SET                    = 0x01,
      MSG_TYPE_REPORT                 = 0x02,
      MSG_TYPE_FORWARD                = 0x03,
      MSG_TYPE_INIT                   = 0x04,
      MSG_TYPE_RESTART                = 0x05,
      MSG_TYPE_PING                   = 0x06,
      /* ... */
      MSG_TYPE_ERROR                  = 0x0E,
      MSG_TYPE_EXTRA                  = 0x0F,
      /* ... */
      MSG_TYPE_ADD_DEVICE             = 0x10,
      MSG_TYPE_DEL_DEVICE             = 0x11,
      MSG_TYPE_ADD_TEST               = 0x12,
      MSG_TYPE_DEL_TEST               = 0x13,
      MSG_TYPE_READ_VALUE             = 0x14,
      MSG_TYPE_ADD_START_ACTION       = 0x15,
      MSG_TYPE_ADD_END_ACTION         = 0x16,
      MSG_TYPE_DEL_ACTION             = 0x17,
      MSG_TYPE_ADD_SCENARIO           = 0x18,
```

```
       MSG_TYPE_DEL_SCENARIO                    = 0x19,
       MSG_TYPE_UPLOAD                          = 0x1A,
       MSG_TYPE_RESUME                          = 0x1B,
       MSG_TYPE_SUSPEND                         = 0x1C,
       MSG_TYPE_CANCEL                          = 0x1D,
       MSG_TYPE_NOTIFY                          = 0x1E,
       MSG_TYPE_SUBSCRIBE_STATUS                = 0x1F,
       MSG_TYPE_UNSUBSCRIBE_STATUS              = 0x20,
       MSG_TYPE_NOTIFY_STATUS                   = 0x21,
}DomoBus_MSG_TYPE;

#ifdef __cplusplus
extern "C" {
#endif
/*****************************************************************************
*
*
*      API    de comunicação - protótipos das funções disponíveis
*
*                                                *
*****************************************************************************/
LINKDLL int DomoBus_SetCommVerbose(int verboseMode);

LINKDLL int DomoBus_GetCommConfig(int *receiveMode,
                                                    int *receiveTimeoutValue,
                                                    int *sendTimeoutValue,
                                                    int *retransmitNum,
                                                    int *retransmitInterval);

LINKDLL int DomoBus_SetCommConfig(int receiveMode,
                                                    int receiveTimeoutValue,
                                                    int sendTimeoutValue,
                                                    int retransmitNum,
                                                    int retransmitInterval);

LINKDLL int DomoBus_CommOpen(DomoBus_ADDRESS myAddress,
                                                DomoBus_COMMHANDLE ComServPort,
                                                DomoBus_COMMHANDLE myPort);

LINKDLL int DomoBus_GetMsg(int *dataLength,
                                            DomoBus_DATA_PTR data,
                                            DomoBus_ADDRESS *originAddress,
                                            DomoBus_ADDRESS *destinationAddress);

LINKDLL int DomoBus_ParseStdMsg(int dataLength,
                                                    DomoBus_DATA_PTR data,
                                                    DomoBus_CTR *controlValue,
                                                    DomoBus_PROPERTY_DESC
*propertyDesc,
                                                    DomoBus_VALUE_PTR
propertyValue,
                                                    BYTE *device);

LINKDLL int DomoBus_DefineReceiveFunction(DomoBus_FUNCTION_PTR receiveFunction,
                                                                    int
dataMaxLength);

LINKDLL int DomoBus_SendMsg(int dataLength,
                                                DomoBus_DATA_PTR data,
                                                DomoBus_ADDRESS originAddress,
                                                DomoBus_ADDRESS
destinationAddress);
```

```
LINKDLL int DomoBus_SendStdMsg(DomoBus_CTR controlValue,
                                                DomoBus_PROPERTY_DESC
propertyDesc,
                                                DomoBus_VALUE_PTR propertyValue,
                                                DomoBus_ADDRESS originAddress,
                                                DomoBus_ADDRESS
destinationAddress);

LINKDLL int DomoBus_SendMsgAsynch(int dataLength,
                                                     DomoBus_DATA_PTR data,
                                                     DomoBus_ADDRESS
originAddress,
                                                     DomoBus_ADDRESS
destinationAddress);

LINKDLL int DomoBus_SendAnswer(int dataLength,
                                                DomoBus_DATA_PTR data,
                                                DomoBus_ADDRESS originAddress,
                                                DomoBus_ADDRESS
destinationAddress);

LINKDLL int DomoBus_SendAnswerError(BYTE errorCode,
                                                DomoBus_ADDRESS originAddress,
                                                DomoBus_ADDRESS
destinationAddress);

LINKDLL int DomoBus_CommClose(void);
/*****************************************************************************/

#ifdef __cplusplus
}
#endif

#endif /* _DOMOBUS_COMM_API_H_ */
```