iCONNECT	**FICHA DE ESPECIFICAÇÃO TÉCNICA**

x PRODUTO	PROCESSO	**DATA: 04/02/2013**

PRODUTO: IcBox	
SUB-PRODUTO:	
DESCRIÇÃO: API DLL c++ (C++ builder e Delphi)	
ELABORAÇÃO: Wagner	ÁREA: PDI

API – DLL ICBOX

Recomendado para uso com C++ Builder e Delphi

//
Métodos disponíveis:

void **initialize**(AnsiString port);
parâmetro: AnsiString port. ex COM1, COM2...
retorno: -
obs: inicializa o componente de comunicação e configura o nome da porta COM. Deve ser utlizado antes de chamar a função openCom()

void **finalize**(void);
parâmetro: -
retorno: -
função: fecha a porta de comunicação (se estiver aberta) e finaliza o componente de comunicação. Deve ser obrigatoriamente chamada antes da aplicação host ser finalizada.

int **openSetupDialog**(char* retBuffer);
parâmetro: ponteiro com a posição que será gravado o retorno
retorno: inteiro com a quantidade de caracteres presente no retorno
função: abre uma janela de diálogo com a configuração para escolha da porta de comunicação com o dispositivo.

int **openCom**(char* retBuffer ,AnsiString porta);
parâmetro 1:ponteiro com a posição que será gravado o retorno
parâmetro 2: String com o nome da porta a ser aberta para comunicação. Caso o parâmetro seja vazio, a operação de abertura se dará com o porta selecionada pelo openSetupDialog ou pela porta default (a primeira disponível, se existir).
retorno: inteiro zero caso a porta seja aberta com sucesso
: inteiro com a quantidade de caracteres do erro caso exista algum erro ou exceção com a falha.
função: abre a porta de comunicação.

boolean **closeCom**(void);
parâmetro: -
retorno : boolean true se porta fechada com sucesso
: boolean false caso exista algum problema - não é informado caso exista erro

função: fecha a porta de comunicação.

int **testCom**(char* retBuffer ,AnsiString arg);
parâmetro 1: ponteiro com a posição que será gravado o retorno
parâmetro 2: String contendo o comando que será enviado pela serial. Necessário colocar caracteres terminadores <Line feed> <Carriage return>
retorno: inteiro com a quantidade de caracteres do retorno com a resposta do dispositivo. A resposta também possui os caracteres terminadores <Line feed> <Carriage return>

boolean **isConnected**(void);
parâmetro: -
retorno: boolean indicando se está conectado ou não.
função: indica se a porta de comunicação está conectada. Esta função confia no estado indicado pelo componente. Para ter certeza de que a comunicação está ativa, utilize a função checkConnectionStatus().

int **getVersionInfo**(char* retBuffer);
parâmetro 1: ponteiro com a posição que será gravado o retorno
retorno: inteiro com a quantidade de caracteres da versão do firmware gravado no dispositivo.
função: solicitar a versão do firmware gravado no dispositivo. Também pode ser utilizado para verificar o funcionamento do canal de comunicação.

boolean **getOnHook**(void);
parâmetro: -
retorno: boolean indicando se o telefone esta no gancho (true) ou fora do gancho(false)
obs: A partir da versão ICB0.36 , este comando retorna o estado mesmo durante uma identificação de ligação sainte.

boolean **checkConnectionStatus**(void);
parâmetro: -
retorno: boolean indicando se a comunicação entre o o programa e o dispositivo está ok. Nesta função um comando é enviado ao dispositivo, caso a resposta recebida seja a esperada, o retorno é (true).

boolean **dialNumber**(AnsiString arg);
parâmetro: String com os números a serem discados, é recomendado utilizar a função getOnHook para verificar se o telefone está fora do gancho e depois solicitar a discagem. O caracter "P" ou "p" aciona uma pausa de 500 ms na discagem.
retorno: boolean com o estado do recebimento do comando. Neste comando, o retorno não ocorre ao fim da execução da discagem e sim ao fim do recebimento do número para discagem
obs: os tons dtmf são gerados por um período de 70ms intercalados por um período de silêncio de 70ms. Existe um limite para 20 dígitos (incluindo a pausa).

AnsiString **getBuffer**(int timeOut);
Função removida. Substituída pela Função getEvent;

int **getEvent**(char* retBuffer, int timeOut);
parâmetro 1: ponteiro com a posição que será gravado o retorno
parâmetro 2: inteiro em milisegundos com o time out da solicitação. Caso seja 0 é utilizado um valor default de 2000 msecs.
retorno: inteiro com a quantidade de caracteres de retorno, caso exista.
obs. com este comando é que é feita a leitura de novos eventos de ligação. o uso desta função deverá ser através de um pooling.

É aconselhado um controle para prevenir o uso deste comando quando se estiver realizando a programação do dispositivo, pois pode ocorrer deste comando "capturar" a resposta de um outro comando.

int **getRXLevel**(char * retBuffer);
parâmetro 1: ponteiro com a posição que será gravado o retorno
retorno: inteiro com a quantidade de caracteres do retorno.
O valor retornado apontado em retBuffer é uma String com o valor do nível de rx gravado no dispositivo. O valor default é 4. O nível com atenuação máxima é 1 e o valor com atenuação mínima 8.
obs. o valor é lido da memória flash do microcontrolador.

bool **setRXLevel**(int arg);
parâmetro: inteiro entre 1 e 8 com nível a ser programado no dispositivo.
retorno: true se a gavação ocorrer com sucesso. false caso acorra algum erro
obs: o valor é gravado na memória flash não volátil. Não é recomendado a gravação periódica (Por ex, a cada inicialização)

int **getRXMode**(char * retBuffer);
parâmetro 1: ponteiro com a posição que será gravado o retorno
retorno: inteiro com o modo de operação gravado no dispositivo.
obs:

bool **setRXMode**(int arg);
parâmetro: inteiro com o modo de operação do dispositivo:
1 - modo automático. o dispositivo aguarda a detecção por dtmf, caso são seja possivel, tenta a detecção por FSK.
2 - DTMF. o dispositivo fará a detecção exclusivamente por DTMF (caso a sinalização DTMF esteja disponível). É indicado caso a detecção automática falhe.
3 - FSK. O dispositivo fará a detecção exclusivamente por FSK (caso a sinalização DTMF esteja disponível). É indicado caso a detecção automática falhe.
obs: o valor é gravado na memória flash não volátil. Não é recomendado a gravação periódica (Por ex, a cada inicialização)

int **getId**(char * retBuffer);
parâmetro: ponteiro com a posição que será gravado o retorno
retorno: inteiro com a quantidade de caracteres retornados. O ponteiro indica a String com 3 posições com valor entre 001 e 999 correspondente ao id do dispositivo
obs:

bool s**etId**(AnsiString arg);
parâmetro: - String com 3 posições com o Id do dispositivo, ex "001", "123"
retorno: booleano com o resultado da configuração
obs:

Atenção Durante uma identificação (entrada ou saida) as funções de configuração não são respondidas pois é dada prioridade à identificação da chamada. No caso de identificação de uma ligação sainte o tempo sem resposta pode ser de até 15 segundos. A exceção são as funções **isConnected e getOnHook.**

Exemplo de declaração (Para mais exemplos, acesse a página de suporte do icbox em www.identech.com.br) :

Borland C++ (utilizando o arquivo .llib):

```
extern "C" __stdcall __declspec(dllimport)AnsiString testCom(AnsiString arg);
```

Em Delphi 7

```
function testCom(arg: String): String ; stdcall; external 'Projeto_dll_v6.dll';
```

HISTÓRICO DE REVISÕES

iCONNECT	**FICHA DE ESPECIFICAÇÃO TÉCNICA**

REVISÃO	DATA	DESCRIÇÃO DA ALTERAÇÃO
0	05/06/2012	Versão Inicial
1	04/02/2013	Mudado formato do documento. Alterado para mais clareza das informações.
2	22/02/2013	Atualização das chamadas de funções.