authotelcom.dll **-** (versão: 0.1.0.5)

Authotelcom.dll foi desenvolvida para a comunicação com o relógio de ponto PointLine 1510. A comunicação pode ser realizada tanto 'Serial' como 'TCP/IP'.

Obs.: *Os exemplos apresentados estão em Delphi5.*

Funções e Procedures:

- configura;
- recebeMarcacoes;
- recebeMarcacoesTCP;
- enviaTrabalhador;
- enviaTrabalhadorTCP;
- enviaEmpregador;
- enviaEmpregadorTCP;
- leEmpregador;
- leEmpregadorTCP;
- leTrabalhador;
- leTrabalhadorTCP;
- backupDigitais;
- backupTCP;
- restauraBackup;
- restauraBackupTCP;
- fecharComunicacao;

Tipos de dados para serem passados como referência nas funções(exceto 'configura'):

```
TMarcacao = record
  nsr: LongWord;
  cont: LongWord;
  pis: string[12];
  dia: byte;
  mes: byte;
  ano: word;
  hora: byte;
  minuto: byte;
end;

TControle = record
  total: Word;
  atual: Word;
  start: boolean;
  erro: Shortint;
  porta: Word;
  s_tipo: byte;
  modelo: byte;
  endereco: string[15];
  backup: ShortString;
  baudrate: integer;
end;
```

```
TDados = record
  adcOUSubst: ShortString;
  pin: ShortString;
  pis: ShortString;
  identificador: ShortString;
  cei: ShortString;
  razaoSocial: ShortString;
  localPrestServ: ShortString;
  tipoId: ShortString;
  nome: ShortString;
  id_bio: ShortString;
  numCartao: ShortString;
  senha: ShortString;
  mestre: ShortString;
  verifica: ShortString;
end;
```

Função 'configura':

Esta função é utilizada para *ajustar hora, ler hora do relógio, ajustar horário de verão e ler horário de verão.*

Declaração: function configura(tipo, ip, host, porta, end_dev, datahorai, datahoraf, diasSem, diasSemF, info: PChar; com, total, atual, duracaoToque, flag, config, qtde_rel, baud: integer): pchar; stdcall; external 'authotelcom.dll';

Definição dos parâmetros:

tipo – (*obrigatório*) indica qual o tipo de dado a ser enviado para o relógio. Os tipos são:
AR – ajuste de relógio;
LR – leitura da hora do relógio;
AH – ajuste do horário de verão;
LH – leitura do horário de verão;

end_dev – (*obrigatório*) – indica para qual endereço os dados serão enviados, podendo ser Unicast, Multicast e Broadcast.
Unicast – para enviar informação para um dispositivo. Ex.: U,00 (U, + endereco do dispositivo)
Multicast – enviar para mais de um dispositivo. Ex.: M,00010203 (M, + enderecos dos dispositivos contando com 2 caracteres cada endereço em sequencia sem separador, os endereços variam de 00 a 31)
Broadcast – enviar para todos os dispositivos. Ex.: B

ip – se comunicação serial, (vazio) ''; se tcp, obrigatório

porta – (*obrigatório*) se serial, 'COM1', por exemplo; se tcp, 1001

host – opcional se tcp; (vazio) '' se serial

datahorai – Data/hora para ajuste no relógio (usado para ajustar relógio e horário de verão sendo nesses dois casos obrigatórios, para as leituras o campo pode ficar vazio)

datahoraf – Data/hora para ajuste no relógio indicando o fim do horário de verão (usado para ajustar horário de verão)

diasSem – (vazio)

diaSemF – (vazio)

info – (vazio)

com – (*obrigatório*) 1 se serial, 2 se tcp

total – 0

atual – 0

duracaoToque – 0

flag – 0

config – 0

qtde_rel – se end_dev igual a B, então quantidade de relógios cadastrados; senão, 0.

baud – (*obrigatório*) Informe a taxa de comunicação (Ex.: 9600)

Obs.: Caso não seja possível estabelecer comunicação com o relógio o retorno será '-1'.

Ajustar Hora:

O formato para o parâmetro *datahorai* é 'dd/mm/yyyy hh:mm:ss'

O retorno da função é o endereço do relógio + 0 ou código de erro*. (O zero indica que a função foi executada com sucesso).

Leitura da Hora

O retorno da função é o endereço do relógio + data/hora no formato 'dd/mm/yyyy hh:mm:ss' ou código de erro*.

Ajustar Horário de Verão:

O formato para o parâmetro *datahorai e datahoraf* é 'dd/mm/yyyy hh', O único item utilizado é a data completa e a hora, dispensando minutos e segundos.

O retorno da função é o endereço do relógio + 0 ou código de erro*. (O zero indica que a função foi executada com sucesso).

Leitura do Horário de Verão

O retorno da função é o endereço do relógio + data/hora de início + data/hora de fim no formato 'dd/mm/yyyy hh:mm dd/mm/yyyy hh:mm' ou código de erro*.

Se o relógio não possuir horário de verão cadastrado o retorno será: endereço do relógio + '00/00/0000 00:00 00/00/0000 00:00'.

Função 'recebeMarcações':

Função utilizada para ler as marcações do relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação serial.

Declaração: function recebeMarcacoes(var marcacao: array of TMarcacao; var controle: Tcontrole; evento: integer): boolean; stdcall; external 'authotelcom.dll';

A primeira chamada da função deverá enviar a variável *controle.start* como *true.*Ao continuar recebendo as marcações a variável passa a ser *false.*

O registro controle.modelo deve indicar qual o modelo do relógio que será feita a comunicação. Sendo: 1 – POINTLine 1510 Card; 2 – POINTLine 1510 BIOProx; 3 – POINTLine 1510 Duo Card e 4 – POINTLine 1510 Duo Card Bio.

Na chamada da função o valor do contador deverá ser passado em *evento* Se o valor for 0 (zero), serão recebidos todas as marcações existentes no relógio. O valor contido em *evento* se refere ao ponto a partir de onde as marcações começarão a ser coletadas.

Todo ponto registrado no relógio contém um número que indica qual é a marcação atual, esse número é salvo na variável *marcacao.cont*, toda vez que for receber evento, o campo *marcacao.nsr*, deverá conter o valor da última marcação recebida do relógio.

Esse valor será passado apenas na primeira chamada da função, onde o *controle.start* será *true*; se a função receber *true* como resposta, indicando que há marcações no relógio, os valores: *marcacao.cont, marcacao.nsr* serão atualizados pela dll, assim como *controle.total* e *controle.atual*.

As variáveis *controle.porta* (com o valor da porta de comunicação; ex.: COM1, então *controle.porta* = 1), *controle.s_tipo* (como a comunicação é serial, *controle.s_tipo* = 1), *controle.endereco* (indica o endereço do relógio que terão suas marcações recebidas) e *controle.baudrate* (padrão - 9600) também deverão ser preenchidas (são campos obrigatórios).

Se o retorno da função for *false* e as variáveis do registro *controle* estiverem nulas, houve falha na comunicação. Se for *false* e a variável *controle.erro* for igual a 96, então não há registros de marcações no relógio.

Se o retorno for *true*, as variáveis do registro *marcacao* terão os valores referentes a marcação atual e o procedimento de chamada da função deverá ser repetido até que o retorno da função seja *false* ou que a variável *controle.total* seja igual a variável *controle*.*atual*.
A função retorna até 10 marcações por vez, dependendo da configuração.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido, os valores podem ser visualizados logo abaixo em *Códigos dos erros**.

Exemplo 1:

Tenho 10 marcações no relógio, na primeira vez em que recebo as marcações envio *marcacao.nsr = 0,* na próxima vez em que chamar a função de receber marcações, envio o *marcacao.nsr = 10*, assim começo a partir da última marcação recebida;

Caso envie *marcacao.nsr = 5,* as marcações serão recebidas a partir do registro de ponto que contém o valor 'contador' igual a 5. Visualizando o exemplo, as últimas cinco marcações seriam

repetidas.

OBS.: Para esta versão da dll, no diretótrio onde se encontrar deverá conter o arquivo *config.rwt* e dentro dele o valor entre 1 e 10. (Valor que indica quantas marcações deverão recebidas por vez)
Esta observação vale para a função recebeMarcacoesTCP.

Função 'recebeMarcaçõesTCP':

Função utilizada para ler as marcações do relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação TCP.

Declaração: function recebeMarcacoesTCP(var marcacao: array of TMarcacao; var controle: Tcontrole; evento: integer): boolean; stdcall; external 'authotelcom.dll';

Essa função difere da função acima apenas no modo de comunicação e nos parâmetros iniciais *controle.s_tipo* e *controle.endereco,* que receberão respectivamente: 2 (valor que indica o tipo comunicação - TCP) e o ip configurado no relógio de ponto.

Função 'enviaTrabalhador':

Função utilizada para enviar dados dos funcionários para o relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação serial.

Declaração: function enviaTrabalhador(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll';

A primeira chamada da função deverá enviar a variável *controle.start* como *true.*Ao continuar enviando os funcionários a variável passa a ser *false.*

O registro controle.modelo deve indicar qual o modelo do relógio que será feita a comunicação. Sendo: 1 – POINTLine 1510 Card; 2 – POINTLine 1510 BIOProx; 3 – POINTLine 1510 Duo Card e 4 – POINTLine 1510 Duo Card Bio.

As variáveis controle.porta (com o valor da porta de comunicação; ex.: COM1, então controle.porta = 1)*, controle.s_tipo* (como a comunicação é serial, *controle.s_tipo = 1*), *controle.endereco* (indica o endereço do relógio que receberá os dados dos funcionários) e *controle.baudrate* (padrão - 9600) também deverão ser preenchidas (são campos obrigatórios). A variável *controle.backup* dever conter 'N'.

As variáveis do registro *dados* deverão ser preenchidas com as informações referentes ao funcionário. Para cada funcionário enviado, o valor da variável *controle.atual* deverá ser incrementada de 1.

A variável *dados.adcOUSubst* terá importância na primeira chamada da função, essa variável indica se os funcionários enviados a seguir serão: adicionados, substituídos, excluídos ou adicionados/substituídos.

> **Adicionar**: para incluir um funcionário no relógio, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'A'**. Nesse caso ele apenas inclui o funcionário na lista do relógio, caso tente enviar um funcionário que possui um mesmo PIS já cadastrado no relógio ou reenviar algum funcionário, haverá um *código de erro** como retorno.

Substituir: para substituir um funcionário já cadastrado no relógio, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'S'.** Nesse caso ele apenas substitui os dados do funcionário cadastrado no relógio pelos dados recebidos. O funcionário é localizado pelo PIS. Caso tente enviar dados de um funcionário cujo PIS inexiste no relógio, haverá um *código de erro** como retorno.

Excluir: para excluir um funcionário do relógio, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'E'**. Nesse caso ele exclui o funcionário na lista do relógio. O funcionário a ser excluído é localizado pelo PIS. Caso tente excluir um funcionário cujo PIS inexiste no relógio, haverá um *código de erro** como retorno.

Adicionar/Substituir: para adicionar/substituir um funcionário no relógio, o comando a ser enviado na variável *dados.adcOUSubst* poderá ser qualquer caracter diferente de **'A'**, **'S'** e **'E'**. Nesse caso ele tenta incluir o funcionário na lista do relógio, se o PIS já estiver cadastrado no relógio, o sistema tenta substituir os dados do funcionário. Se não for possível executar nenhuma das duas opções, haverá um *código de erro** como retorno.

Os campos a serem preenchidos no registro *dados* com as informações dos funcionários (todos os campos referentes às informações dos funcionários são obrigatórios, exceto o campo *dados.id_bio*):

dados.adcOUSubst – indica qual o tipo de ação a ser executada (definida acima).
dados.pin – número de identificação do funcionário. (código, matrícula...).
dados.pis – número do pis do funcionário.

dados.identificador – usado no envio de empregador.
dados.cei – usado no envio de empregador.
dados.razaoSocial – usado no envio de empregador.
dados.localPrestServ – usado no envio de empregador.
dados.tipoId – usado no envio de empregador.

dados.nome – nome do funcionário (máximo de 52 caracteres).
dados.id_bio – sempre 0 (controle do firmware).
dados.numCartao – número do cartão de proximidade ou código de barras.
dados.senha – senha numérica, máximo de 6 dígitos. (Campo opcional, porém caso seja cadastrado deve ter no mínimo 3 dígitos)
dados.mestre – indica se o funcionário é mestre (0 – funcionário comum / 1 – funcionário mestre)
dados.verifica – campo não utilizado nessa versão.

Se o retorno for *true*, os dados do próximo funcionário poderão ser enviados para o relógio. Caso o retorno da função seja *false,* houve falha na comunicação.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido, os valores podem ser visualizados logo abaixo em *Códigos dos erros**.

Função 'enviaTrabalhadorTCP':

Função utilizada para enviar dados dos funcionários para o relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação TCP.

Declaração: function enviaTrabalhadorTCP(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll';

Essa função difere da função acima apenas no modo de comunicação e nos parâmetros iniciais *controle.s_tipo* e *controle.endereco,* que receberão respectivamente: 2 (valor que indica o tipo comunicação - TCP) e o ip configurado no relógio de ponto.

Função 'enviaEmpregador':

Função utilizada para enviar dados do empregador para o relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação serial.

Declaração: function enviaEmpregador(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll':

O relógio de ponto poderá receber apenas um empregador. Portanto essa função será chamada apenas duas vezes. (Uma chamada, com o *controle.start* = *true*, indicando que será feito o envio do empregador e uma segunda chamada, *controle.start* = *false*, enviando os dados do empregador).

O registro controle.modelo deve indicar qual o modelo do relógio que será feita a comunicação. Sendo: 1 – POINTLine 1510 Card; 2 – POINTLine 1510 BIOProx; 3 – POINTLine 1510 Duo Card e 4 – POINTLine 1510 Duo Card Bio.

As variáveis controle.porta (com o valor da porta de comunicação; ex.: COM1, então controle.porta = 1)*, controle.s_tipo* (como a comunicação é serial, *controle.s_tipo = 1*), *controle.endereco* (indica o endereço do relógio que receberá os dados do empregador) e *controle.baudrate* (padrão - 9600) também deverão ser preenchidas (são campos obrigatórios). Assim como *controle.atual* e *controle.total* que deverão ser mantidas em 1.

As variáveis do registro *dados* deverão ser preenchidas com as informações referentes ao empregador.

A variável *dados.adcOUSubst* terá importância na primeira chamada da função, essa variável indica se o empregador enviado a seguir será: adicionado, substituído, excluído ou adicionado/substituído.

> **Adicionar**: para incluir o empregador no relógio, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'A'**. Nesse caso ele apenas inclui o empregador no relógio, caso já exista uma empresa cadastrada no relógio, haverá um *código de erro** como retorno.
>
> **Substituir**: para substituir o empregador já cadastrado no relógio, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'S'.** Nesse caso ele apenas substitui os dados do empregador cadastrado no relógio pelos dados recebidos. A empresa é localizada pelo CNPJ/CPF. Caso tente enviar dados onde o CNPJ/CPF inexiste no relógio, haverá um *código de erro** como retorno.
>
> **Adicionar/Substituir**: para adicionar/substituir o empregador no relógio, o comando a ser enviado na variável *dados.adcOUSubst* poderá ser qualquer caracter diferente de **'A'** e **'S'**. Nesse caso ele tenta incluir a empresa no relógio, se o CNPJ/CPF já estiver cadastrado no relógio, o sistema tenta substituir os dados do empregador. Se não

for possível executar nenhuma das duas opções, haverá um *código de erro** como retorno.

Os campos a serem preenchidos no registro *dados* com as informações da empresa (todos os campos referentes às informações dos funcionários são obrigatórios, exceto o campo *dados.cei*):

dados.adcOUSubst – indica qual o tipo de ação a ser executada (definida acima).

dados.pin – usado no envio de funcionários.
dados.pis – usado no envio de funcionários.

dados.identificador – número do CNPJ ou do CPF.
dados.cei – número do CEI, caso não haja, enviar '0'.
dados.razaoSocial – razão social da empresa (máximo de 150 caracteres).
dados.localPrestServ – endereço da empresa (local de prestação de serviço - máximo de 100 caracteres).
dados.tipoId – informa qual o tipo de identificador a ser enviado (1 – CNPJ / 2 – CPF).

dados.nome – usado no envio de funcionários.
dados.id_bio – usado no envio de funcionários.
dados.numCartao – usado no envio de funcionários.
dados.senha – usado no envio de funcionários.
dados.mestre – usado no envio de funcionários.
dados.verifica – usado no envio de funcionários.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido, os valores podem ser visualizados logo abaixo em *Códigos dos erros**.

Obs.: Após o envio do empregador o mesmo não poderá ser apagado do relógio, podendo apenas que seus dados sejam substituídos.

Função 'enviaEmpregadorTCP':

Função utilizada para enviar dados do empregador para o relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação TCP.

Declaração: function enviaEmpregadorTCP(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll';

Essa função difere da função acima apenas no modo de comunicação e nos parâmetros iniciais *controle.s_tipo* e *controle.endereco,* que receberão respectivamente: 2 (valor que indica o tipo comunicação - TCP) e o ip configurado no relógio de ponto.

Função 'leEmpregador':

Função utilizada para ler dados do empregador do relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação serial.

Declaração: function leEmpregador(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll';

O relógio de ponto contém apenas um empregador cadastrado e essa função será chamada duas vezes. (Primeira chamada, *controle.start = true*, indicando que será feita a leitura do empregador, na segunda chamada, *controle.start = false*, confirmação dos dados).

O registro controle.modelo deve indicar qual o modelo do relógio que será feita a comunicação. Sendo: 1 – POINTLine 1510 Card; 2 – POINTLine 1510 BIOProx; 3 – POINTLine 1510 Duo Card e 4 – POINTLine 1510 Duo Card Bio.

As variáveis *controle.porta* (com o valor da porta de comunicação; ex.: COM1, então controle.porta = 1)*, controle.s_tipo* (como a comunicação é serial, *controle.s_tipo = 1*), *controle.endereco* (indica o endereço do relógio que receberão os dados do empregador) e *controle.baudrate* (padrão - 9600) também deverão ser preenchidas (são campos obrigatórios).

Os dados recebidos do relógio se encontrarão nas variáveis: *dados.identificador, dados.cei, dados.razaoSocial, localPrestServ, dados.tipoId*.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido, os valores podem ser visualizados logo abaixo em *Códigos dos erros**.

Função 'leEmpregadorTCP':

Função utilizada para ler dados do empregador do relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação TCP.

Declaração: function leEmpregadorTCP(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll';

Essa função difere da função acima apenas no modo de comunicação e nos parâmetros iniciais *controle.s_tipo* e *controle.endereco,* que receberão respectivamente: 2 (valor que indica o tipo comunicação - TCP) e o ip configurado no relógio de ponto.

Função 'leTrabalhador':

Função utilizada para ler dados dos funcionários cadastrados no relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação serial.

Declaração: function leTrabalhador(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll';

As variáveis controle.porta (com o valor da porta de comunicação; ex.: COM1, então controle.porta = 1)*, controle.s_tipo* (como a comunicação é serial, *controle.s_tipo = 1*), *controle.endereco* (indica o endereço do relógio que receberá os dados do trabalhador) e *controle.baudrate* (padrão - 9600) também deverão ser preenchidas (são campos obrigatórios).

O registro controle.modelo deve indicar qual o modelo do relógio que será feita a comunicação. Sendo: 1 – POINTLine 1510 Card; 2 – POINTLine 1510 BIOProx; 3 – POINTLine 1510 Duo Card e 4 – POINTLine 1510 Duo Card Bio.

Os dados recebidos do relógio se encontrarão nas variáveis: *dados.pin, dados.pis, dados.nome, dados.id_bio*, *dados.numCartao*, *dados.senha*, *dados.mestre*.

O campo *dados.senha*, retornará o valor '000000', quando não houver senha cadastrada.
A senha é numérica, caso haja zeros à esquerda, o valor retornado será 'A'. A senha possui o máximo de 6 dígitos.

Ex.: A senha enviada para o relógio ou digitada no mesmo é: 00123. Nesse caso se for feita a leitura do trabalhor, o retorno no campo senha será: 0AA123, os zeros a esquerda devem ser desconsiderados, pois são apenas complementos para o campo de tamanho 6. As letras 'A's' indicam a quantidade de zeros à esquerda.
Se digitado 123, retorno será 000123; se digitado 1020, retorno será 001020. Se digitado 012305, retorno A12305.

Os campos *controle.atual* e *controle.total* serão atualizados pela dll, para cada informação recebida a função deve ser chamada novamente para que seja feita a confirmação dos dados recebidos. Caso haja mais dados, em cada confirmação os campos do registro *dados* estarão com informações do próximo funcionário, até que *controle.atual* seja igual a *controle.total* encerrando o ciclo com uma confirmação.

O campo *dados.pis* pode ser preenchido e enviado para o relógio, nesse caso, o relógio envia os dados somente do funcionário com o PIS enviado. Caso contrário, o *dados.pis* deve ser igual a 0 (zero), assim todos os funcionários cadastrados no relógio serão lidos.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido, os valores podem ser visualizados logo abaixo em *Códigos dos erros**.

Função 'leTrabalhadorTCP':

Função utilizada para ler dados dos funcionários cadastrados no relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação TCP.

Declaração: function leTrabalhadorTCP(var dados: TDados; var controle: Tcontrole): boolean; stdcall; external 'authotelcom.dll';

Essa função difere da função acima apenas no modo de comunicação e nos parâmetros iniciais *controle.s_tipo* e *controle.endereco,* que receberão respectivamente:2 (valor que indica o tipo comunicação - TCP) e o ip configurado no relógio de ponto.

Função 'backupDigitais':

Obs.: Função utilizada apenas para os relógio de modelo POINTLine 1510 Duo Card Bio.

Função utilizada para fazer o backup das digitais do relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência.
Os erros podem ser vistos abaixo em Erros backup**.

Declaração: function backupDigitais(var controle: Tcontrole):boolean; stdcall; external 'authotelcom.dll';

As variáveis *controle.porta* (com o valor da porta de comunicação; ex.: COM1, então controle.porta = 1)*, controle.s_tipo* (como a comunicação é serial, *controle.s_tipo = 1*), *controle.endereco* (indica o endereço do relógio que será feito o backup das digitais) e

controle.baudrate (padrão - 9600) deverão ser preenchidas (são campos obrigatórios).
A variável *controle.backup* deverá conter 'S'.

O registro controle.modelo deve indicar qual o modelo do relógio que será feita a comunicação. Sendo: 1 – POINTLine 1510 Card; 2 – POINTLine 1510 BIOProx; 3 – POINTLine 1510 Duo Card e 4 – POINTLine 1510 Duo Card Bio.

A dll criará uma pasta chamada backup onde salvará as digitais com o nome dd_mm_yy hhnnss.ezm.

Ex.: Backup realizado, o nome do arquivo seria: 14_09_20 173840.ezm

Função 'backupTCP':

Função utilizada para fazer o backup das digitais do relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação TCP.

Declaração: function backupTCP(var controle: Tcontrole):boolean; stdcall; external 'authotelcom.dll';

Essa função difere da função acima apenas no modo de comunicação e nos parâmetros iniciais *controle.s_tipo* e *controle.endereco,* que receberão respectivamente: 2 (valor que indica o tipo comunicação - TCP) e o ip configurado no relógio de ponto.

Os erros podem ser vistos abaixo em Erros backup**.

Função 'restauraBackup':

Obs.: Função utilizada apenas para os relógio de modelo POINTLine 1510 Duo Card Bio.

Função utilizada para fazer restaurar o backup das digitais no relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência.
Os erros podem ser vistos abaixo em Erros backup**.

Declaração: function restauraBackup(var controle: Tcontrole; caminho: pchar):boolean; stdcall; external 'authotelcom.dll';

As variáveis *controle.porta* (com o valor da porta de comunicação; ex.: COM1, então controle.porta = 1)*, controle.s_tipo* (como a comunicação é serial, *controle.s_tipo = 1*), *controle.endereco* (indica o endereço do relógio que será feito o backup das digitais) e *controle.baudrate* (padrão – 9600) deverão ser preenchidas (são campos obrigatórios).
A variável *controle.backup* deverá conter 'S'.

O registro controle.modelo deve indicar qual o modelo do relógio que será feita a comunicação. Sendo: 1 – POINTLine 1510 Card; 2 – POINTLine 1510 BIOProx; 3 – POINTLine 1510 Duo Card e 4 – POINTLine 1510 Duo Card Bio.

No parâmetro *caminho* deve ser indicado o caminho do arquivo .ezm a ser restaurado.

Função 'restauraBackupTCP':

Obs.: Função utilizada apenas para os relógio de modelo POINTLine 1510 Duo Card Bio.

Função utilizada para fazer restaurar o backup das digitais no relógio de ponto. O retorno se encontra nos parâmetros que são passados por referência. Função para comunicação TCP.
Os erros podem ser vistos abaixo em Erros backup**.

Declaração: function restaurabackupTCP(var controle: Tcontrole; caminho: pchar):boolean; stdcall; external 'authotelcom.dll';

Essa função difere da função acima apenas no modo de comunicação e nos parâmetros iniciais *controle.s_tipo* e *controle.endereco,* que receberão respectivamente: 2 (valor que indica o tipo comunicação - TCP) e o ip configurado no relógio de ponto.

Procedure 'fecharComunicação':

Obs.: Procedure utilizada apenas para a comunicação TCP/IP.

Procedure utilizada para interromper/finalizar a comunicação com o relógio de ponto.

Declaração: procedure fecharComunicacao; stdcall; external 'authotelcom.dll';

***Códigos dos Erros:**

01	BCC recebido não confere com BCC calculado.
02	Hora desejada (enviada pelo PC) não constitui uma hora válida.
03	Parâmetro e/ou Tamanho e/ou Flag/Error não suportados.
04	O Comando enviado não é suportado ou é desconhecido.
05	Erro não especificado.
08	Não foi encontrado um funcionário com o PIS solicitado.
10	Identificador (cpf/cnpj/pis) inconsistente.
11	Identificador (cpf/cnpj/pis) recusado.
12	Código recusado.
13	Espaço insuficiente.
96	O Frame recebido contém erro.

****Erros backup:**

0	Sucesso.
-1	Não foi possível abrir a porta serial.
-3	Não foi possível escrever na porta serial.
-5	Não foi possível ler a porta serial
-6	Sem resposta do relógio.
-7	Resposta incorreta.
-101	Falha de leitura.
-102	Usuário não encontrado.
-104	Tente novamente.
-105	Sem resposta.
-106	Memória esgotada.
-107	Usuário já existente.
-108	Limite de dedos atingido.
-110	Usuário inválido.
-112	Ocupado.
-113	Cancelado.
-115	Dedo existente.
-118	Módulo travado.
-203	Arquivo inválido.

Rw Tecnologia Ind. E Com. Ltda
Centro Empresarial Paulo Frederico de Toledo, 80A
Bairro Arco-Íris
Santa Rita do Sapucaí – MG
Telefone:(35)3471-3172
FAX: (35)3471-3353
e-mail: rw@rwtech.com.br
Home Page: www.relogio1510.com.br