TAU
Aurora Technologies' Time & Attendance Unit

TAU Communication API
TAU100.DLL

Aurora Technologies Ltd.
P.O.Box 388, Tirat Carmel 30200, Israel
Tel: (972) 4-8576982, FAX: (972) 4-8576983
http://www.aurora.co.il

Document Version 1.01 January 2000

Table of Contents

Communication Settings

Communication Initializing and Teminating

Basic Communication Command

Time & Attandance Unit API - I/O Level

Continue ...

Time & Attandance Unit API - Application Level

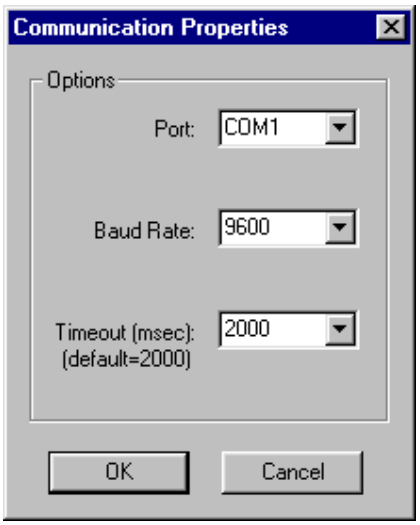
Time & Attandance Card API - Card Level

TAU_SetCommProperties

```
void TAU_SetCommProperties(HWND hwndParent);
```

Description

Set the communication properties using a user interface dialog box. The user can modify the communication port, the timeout value and the baud rate.



Input Parameters

`HWND hwndParent`

The parent window that called the dialog box.
If the value of `hwndParent` equals zero, the desktop is the owner of the dialog box.

Output Parameters

None.

Returns

None.

Example

```
TAU_SetCommProperties(hwndParent);
```

TAU_GetCommPort

```
WORD TAU_GetCommPort(void);
```

Description

Returns the current communication port used by the user application. The initial value is taken from the *TAU100.INI* file in the windows directory.
The user application can modify its own port by calling the *TAU_SetCommPort* function.

The *TAU100.INI* file will look like this:

```
[Comm]
Port=1
Timeout=2000
```

Input Parameters

None.

Output Parameters

None.

Returns

```
WORD nCommPort;
```

The current communication port:
1 for COM1
2 for COM2 etc.

Default value is 1 (when no *TAU100.INI* file exists).

Example

```
int nCommPort;

nCommPort = TAU_GetCommPort();
printf("The communication port in use is COM%d \n",nCommPort);
```

TAU_SetCommPort

```
void TAU_SetCommPort(WORD nCommPort);
```

Description

Set the current communication port used by the user application. The input value does not change the value in the *TAU100.INI* file (exists in the windows directory). The user application can modify this value by calling to *TAU_SetDefaultCommPort* function.

Input Parameters

`WORD nCommPort`	The proper communication port: 1 for COM1 2 for COM2 etc.

Output Parameters

None.

Returns

None.

Example

```
int nCommPort = 2; //COM2

TAU_SetCommPort(nCommPort);
nCommPort = TAU_GetCommPort();
printf("The communication port in use is COM%d \n",nCommPort);
```

TAU_GetCommTimeout

```
DWORD TAU_GetCommTimeout(void);
```

Description

Returns the current communication timeout value used by the user application. Initial value is taken from the *TAU100.INI* file in the windows directory.
The user application can modify its own timeout value by calling the *TAU_SetCommTimeout* function.

The *TAU100.INI* file will look like this:

```
[Comm]
Port=1
Timeout=2000
```

Input Parameters

None.

Output Parameters

None.

Returns

```
DWORD nTimeout;
```

Returns timeout value in millisecond units (e.g. 1000 milli-seconds is 1 second).
Default value is 2000 milliseconds, representing 2 seconds (when no *TAU100.INI* file exists).

Example

```
DWORD nTimeout;

nTimeout = TAU_GetCommTimeout();
printf("The communication timeout in use is");
printf("%d milli-seconds \n",nTimeout);
```

TAU_SetCommTimeout

```
void TAU_SetCommTimeout(DWORD nTimeout);
```

Description

Set the current communication timeout used by the user application. The input value does not change the value in the *TAU100.INI* file (exists in the windows directory). The user application can modify this value by calling the *TAU_SetDefaultCommTimeout* function.

Input Parameters

`DWORD nTimeout`	The timeout in millisecond units. 1000 milli-second is equal to 1 second. A value of 2000 milli-seconds is recommended. A value of 1000 milli-seconds can also be considered.

Output Parameters

None.

Returns

None.

Example

```
DWORD nTimeout;  //milli-seconds

nTimeout = TAU_GetCommTimeout();
if (nTimeout < 1000)
{
   nTimeout = 2000;
   TAU_SetCommTimeout(nTimeout);
}
```

TAU_GetCommBaud

```
DWORD TAU_GetCommBaud(void);
```

Description

Returns the current communication baud rate value used by the user application. The initial value is taken from the *TAU100.INI* file in the windows directory.
The user application can modify its own baud rate value by calling the *TAU_SetCommBaud* function.

TAU100.INI file will look like this:

```
[Comm]
Port=1
Timeout=2000
Baud=9600
```

Input Parameters

None.

Output Parameters

None.

Returns

```
DWORD nBaud;
```

Returns baud rate value. Currently, the baud rate for POWERTIME, version 1.03 and up, should be 9600 baud.

Example

```
DWORD nBaud;

nBaud = TAU_GetCommBaud();
printf("The communication baud rate in use is");
printf("%d\n",nBaud);
```

TAU_SetCommBaud

```
void TAU_SetCommBaud(DWORD nBaud);
```

Description

Set the current communication baud rate to be used by the user application. The input value does not change the value in the *TAU100.INI* file (which exists in the windows directory). The user application can modify this value by calling the *TAU_SetDefaultCommBaud* function. Currently, the value for POWERTIME version 1.03 and up should be 9600 baud.

Input Parameters

`DWORD nBaud`	The baud rate to use. Value should be 9600.

Output Parameters

None.

Returns

None.

Example

```
DWORD nBaud;

nBaud = TAU_GetCommBaud();
if (nBaud != 9600)
{
   nBaud = 9600;
   TAU_SetCommBaud(nBaud);
}
```

TAU_GetDefaultCommPort

```
WORD TAU_GetDefaultCommPort(void);
```

Description

Returns the default communication port that should be used by all applications. Value is taken from the *TAU100.INI* file in the windows directory.
The user application can modify its own port by calling the *TAU_SetCommPort* function.

TAU100.INI file will look like this:

```
[Comm]
Port=1
Timeout=2000
Baud=9600
```

Input Parameters

None.

Output Parameters

None.

Returns

```
WORD nCommPort;
```

The default communication port:
1 for COM1
2 for COM2 etc.

Default value is 1 (when no *TAU100.INI* file exists).

Example

```
int nCommPort;

nCommPort = TAU_GetDefaultCommPort();
printf("The default communication port in use is COM%d \n",
       nCommPort);
```

TAU_SetDefaultCommPort

```
void TAU_SetDefaultCommPort(WORD nCommPort);
```

Description

Set the default communication port used by all applications. The input value changes the value in the *TAU100.INI* file (that exists in the windows directory). The user application can modify its own value by calling the *TAU_SetCommPort* function.

TAU100.INI file will look like this:

```
[Comm]
Port=1
Timeout=2000
Baud=9600
```

Input Parameters

`WORD nCommPort`	The proper communication port: 1 for COM1 2 for COM2 etc.

Output Parameters

None.

Returns

None.

Example

```
int nCommPort = 2; //COM2

TAU_SetDefaultCommPort(nCommPort); //change port in INI file
nCommPort = TAU_GetDefaultCommPort();
printf("The communication port in use is COM%d \n",nCommPort);
```

TAU_GetDefaultCommTimeout

```
DWORD TAU_GetDefaultCommTimeout(void);
```

Description

Returns the current communication timeout value used by all applications. The value is taken from the *TAU100.INI* file in the windows directory.
The user application may modify its own timeout value by calling the *TAU_SetCommTimeout* function.

TAU100.INI file will look like this:

```
[Comm]
Port=1
Timeout=2000
Baud=9600
```

Input Parameters

None.

Output Parameters

None.

Returns

```
DWORD nTimeout;
```

Returns the timeout value in milli-second units (e.g. 1000 milli-seconds is 1 second). The default value is 2000 milli-seconds, equal to 2 seconds (when no *TAU100.INI* file exists).

Example

```
DWORD nTimeout;

nTimeout = TAU_GetDefaultCommTimeout();
printf("The communication timeout in use is");
printf("%d milli-seconds \n",nTimeout);
```

TAU_SetDefaultCommTimeout

```
void TAU_SetDefaultCommPort(DWORD nTimeout);
```

Description

Set the default communication timeout used by all applications. The input value changes the value in the *TAU100.INI* file (that exists in the windows directory). The user application may modify this value by calling the *TAU_SetDefaultCommTimeout* function.

Input Parameters

`DWORD nTimeout`	The timeout in milli-second units. 1000 milli-seconds equal 1 second. A value of 2000 milli-seconds is recommended. A value of 1000 milli-seconds may also be considered.

Output Parameters

None.

Returns

None.

Example

```
DWORD nTimeout;  //milli-seconds

nTimeout = TAU_GetDefaultCommTimeout();
if (nTimeout < 1000)
{
   nTimeout = 2000;
   TAU_SetDefaultCommTimeout(nTimeout);
}
```

TAU_GetDefaultCommBaud

```
DWORD TAU_GetDefaultCommBaud(void);
```

Description

Returns the current communication baud rate value used by all applications. The value is taken from the *TAU100.INI* file in the windows directory.
The user application may modify its own baud rate value by calling the *TAU_SetCommBaud* function.

The *TAU100.INI* file will look like this:

```
[Comm]
Port=1
Timeout=2000
Baud=9600
```

Input Parameters

None.

Output Parameters

None.

Returns

```
DWORD nBaud;
```

Returns the value of the baud rate. The default value should be 9600.

Example

```
DWORD nBaud;

nTimeout = TAU_GetDefaultCommBaud();
printf("The communication baud in use is");
printf("%d \n",nBaud);
```

TAU_GetDefaultCommTimeout

```
void TAU_GetDefaultCommPort(DWORD nBaud);
```

Description

Set the default communication baud used by all applications. The input value changes the value in the *TAU100.INI* file (exists in the windows directory). The user application can modify this value by calling to *TAU_SetDefaultCommBaud* function. Default value should be 9600.

Input Parameters

`DWORD nBaud`	The baud rate. Value should be 9600.

Output Parameters

None.

Returns

None.

Example

```
DWORD nBaud;

nBaud = TAU_GetDefaultCommBaud();
if (nBaud < 9600)
{
   nBaud = 9600;
   TAU_SetDefaultCommBaud(nBaud);
}
```

TAU_CommOpen

```
void TAU_CommOpen(BYTE IsDebug);
```

Description

Open the application service to the TAU. User application can open service with a debug window, that will show string commands in that debug window.
This function does not open the communication port. The communication port is open by the function *TAU_CommConnect* .

Input Parameters

`BYTE IsDebug` — Is debug mode in use:
0 No debug.
1 Debug mode. A debug window will be open.

Output Parameters

None.

Returns

None.

Example

```
BYTE IsDebug = FALSE;
WORD IsCommOK = 0;

//begin
TAU_CommOpen(IsDebug);
IsCommOK = TAU_CommConnect();

//do some actions
TAU_Beep(400); //milli-seconds
TAU_LcdClear();

//end
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_CommClose

```
void TAU_CommClose(void);
```

Description

Close the application service to the TAU. This function does not close the communication port. Function *TAU_CommDisconnect* should be called to close the port.

Input Parameters

None.

Output Parameters

None.

Returns

None.

Example

```
BYTE IsDebug = FALSE;
WORD IsCommOK = 0;

//begin application
TAU_CommOpen(IsDebug);
IsCommOK = TAU_CommConnect();

//do some actions
TAU_Beep(400); //milli-seconds
TAU_LcdClear();

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_CommConnect

```
WORD TAU_CommConnect();
```

Description

Open the communication port. This function should be called after calling *TAU_CommOpen.*

Input Parameters

None.

Output Parameters

None.

Returns

```
WORD IsCommOK;  //0=Communication error, 1=Comm OK
```

Returns the communication status with the TAU. If 0 is returned, the communication port can not be opened or used.

Example

```
BYTE IsDebug = FALSE;
WORD IsCommOK = 0;

//begin application
TAU_CommOpen(IsDebug);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//do some actions
TAU_Beep(400); //milli-seconds
TAU_LcdClear();

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_CommDisconnect

```
WORD TAU_CommDisconnect();
```

Description

Close the communication port. This function should be called before calling *TAU_CommClose.*

Input Parameters

None.

Output Parameters

None.

Returns

```
WORD IsCommOK;  //0=Communication error, 1=Comm OK
```

Returns the communication status with the TAU. If 0 is returned, the communication port can not be closed.

Example

```
BYTE IsDebug = FALSE;
WORD IsCommOK = 0;

//begin application
TAU_CommOpen(IsDebug);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//do some actions
TAU_Beep(400); //milli-seconds
TAU_LcdClear();

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_CommCommand

```
void TAU_CommCommand(BYTE * bpBufferWrite,
                     WORD nLengthToWrite,
                     BYTE * bpBufferRead,
                     WORD nLengthToRead,
                     WORD * nLengthRead);
```

Description

Write a command to the TAU, and read the response from the TAU. This is the basic function used by almost all *TAU_* functions in the POWERTIME API (e.g. *TAU_MemWrite*).
Buffers (read and write) contain the pre-defined protocol strings.

This function deals with the communication port and timeout that were predefined by the user application or by the default values in the *TAU100.INI* file.

Input Parameters

`BYTE * bpBufferWrite`	The input command to write to the TAU.
`WORD nLengthToWrite`	The length (total bytes) of the command buffer to write to the TAU.
`WORD nLengthToRead`	The maximum length of the return buffer from the TAU. A value of 100 is enough.

Output Parameters

`BYTE * bpBufferRead`	The output response buffer read from the POWERTIME unit.
`WORD * nLengthRead`	The length of the response buffer actually read from the POWERTIME unit.

Returns

```
WORD IsCommOK;  //0=Communication error, 1=Comm OK
```

Returns the communication status with the POWERTIME unit. If 0 is returned, communication with the POWERTIME unit failed (wrong comm port, POWERTIME is not in a communication mode). If returns 1, communication succeeded.

Example

```
BYTE bpBufferWrite[] = "T1200400\r\n"; //beep for 400ms
WORD nLengthToWrite = strlen(bpBufferWrite);
BYTE bpBufferRead[100];
WORD nLengthToRead = sizeof(bpBufferRead);
WORD nLengthRead;
WORD IsCommOK;
```

```
//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//do beep action (the hard way)
IsCommOK = TAU_CommCommand(bpBufferWrite,
                           nLengthToWrite,
                           bpBufferRead,
                           nLengthToRead,
                           &nLengthRead);
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
//response should be "T1\r\n"

//do beep action again (the simple way)
IsCommOK = TAU_Beep(400);
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_IsCardInserted

```
WORD TAU_IsCardInserted(BYTE * Result);
```

Description

Asks the POWERTIME unit if a card is present in the reader.

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 No card is present (no card was inserted)
1 A card is present (a card was inserted)

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;

//begin appli
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

IsCommOK = TAU_IsCardInserted(&Result);
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
if (Result == 0)
{
   printf("No card is present...\n");
   return;
}
printf("Card is present ...\n");
//read card data ...

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_CardReset

```
WORD TAU_CardReset(BYTE * Result,
                   BYTE * ChipType);
```

Description

Reset the card and return the type of the chip.

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
Always return 1.

`BYTE * ChipType`

The chip type (known values):
00 Unknown card type or card is not inserted
34 (equal to 22H in hex notation) -- 2K byte memory card.
40 (equal to 28H in hex notation) -- 8K byte memory card.

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
BYTE ChipType;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
IsCommOK = TAU_CardReset(&Result,&ChipType);
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
if (Result == 1 && ChipType != 0)
{
   printf("Card type is %d\n",ChipType);
   return;
}
//end application
TAU_CommDisconnect();
```

```
TAU_CommClose();
```

TAU_CardRead

```
WORD TAU_CardRead(DWORD Offset,
                  BYTE Length,
                  BYTE * Result,
                  BYTE * Buffer);
```

Description

Reads data from the card, starting from a given offset, of a given length.

Input Parameters

`DWORD Offset`

Offset address:
2K cards: 0 - 2047
8K cards: 0 - 8191

`BYTE Length`

Length to read:
1 - 20 (length value is limited due to the protocol).

Output Parameters

`BYTE * Result`

The output result:
0 No card is present (no card was inserted)
1 A card is present (a card was inserted)

`BYTE * Buffer`

The output buffer

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
BYTE Result, CardType, Length;
WORD IsCommOK, Offset;
BYTE Buffer[100];

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
```

```
IsCommOK = TAU_CardReset(&Result,&CardType);
if (IsCommOK && Result != 0) //comm ok, card is present
{
   Offset = 0;
   Length = 16;
   IsCommOK = TAU_CardRead(Offset,Length,&Result,Buffer);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_CardWrite

```
WORD TAU_CardWrite(DWORD Offset,
                   BYTE Length,
                   const BYTE * Buffer,
                   BYTE * Result);
```

Description

Writes data to the card, starting from a given offset, of a given length.

Input Parameters

`DWORD Offset`	Offset address: 2K cards: 0 - 2047 8K cards: 0 - 8191
`BYTE Length`	Length to read: 1 - 20 (length value is limited due to the protocol).
`BYTE * Buffer`	The input buffer to write

Output Parameters

`BYTE * Result`	The output result: 0 No card is present (no card was inserted) 1 A card is present (a card was inserted)

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Result, CardType, Length;
WORD IsCommOK, Offset;
BYTE Buffer[] = "ABCDEFGHIJ";

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
```

```
IsCommOK = TAU_CardReset(&Result,&CardType);
if (IsCommOK && Result != 0) //comm ok, card is present
{
   Offset = 0;
   Length = sizeof(Buffer);
   IsCommOK = TAU_CardWrite(Offset,Length,Buffer,&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_Beep

```
WORD TAU_Beep(WORD Interval);
```

Description

Sound the POWERTIME unit buzzer for a period of time of length Interval.

Input Parameters

`WORD Interval` — Time Interval in milliseconds.

Warning:
If the interval value is larger than or close to the predefined communication timeout period, a communication timeout may occur.

Output Parameters

None

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

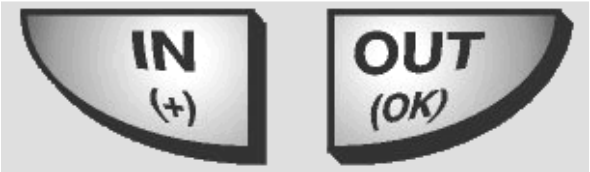
```
WORD IsCommOK;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
TAU_SetCommTimeout(500);
IsCommOK = TAU_Beep(1000); //beep for one second
if (!IsCommOK) //comm not ok, timeout (must happen)
{
   printf("Timeout ...\n");
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_KeyRead

```
WORD TAU_KeyRead(BYTE * Result, BYTE * Key);
```

Description

Read a pressed key from the TAU. Currently, this function should not be applied [the POWERTIME unit itself checks if a key was pressed in order to terminate the communication (OUT key)]. Known keys are **IN**, **OUT** and **TASK**.



Input Parameters

None

Output Parameters

`BYTE * Result`

The output result:
0 No key was pressed.
1 A key was pressed.

`BYTE * Key`

The pressed key value:
0 no key was pressed.
1 IN [+] key was pressed.
2 OUT [ok] key was pressed.
4 TASK key was pressed (hidden key).

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result, Key = 0;
char * KeyName[] = {“None”,”IN”,”OUT”,”IN & OUT”,”TASK”};

TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf(“Communication error ...\n”);
   return;
}
```

```
IsCommOK = TAU_KeyRead(&Result,&Key);
if (IsCommOK && Key != 0) //comm ok, key was pressed
  printf("Key %s was pressed.\n",KeyName[Key]);

TAU_CommDisconnect();
TAU_CommClose();
```

TAU_LcdError

```
WORD TAU_LcdError(BYTE Error);
```

Description

Print an error message on the POWERTIME unit LCD display. The format of the message is "Error XXX " where XXX represent the error number.

Input Parameters

`WORD Error`

Error number 0 - 99.

Note:
The POWERTIME unit has an internal list of known errors. For example, "Error 052" indicates an error in reading the card.

Known values:

POWERTIME Unit errors (01 - 29)
01 unknown unit error
11 memory full (can not append transaction)
22 can not read memory
23 can not write memory

Card Errors (50..59) - I/O Errors
51 unknown card.
52 can not read card
53 can not write card

Card Errors (60..69) - Application Errors
60 invalid project code
61 invalid card version
62 invalid chksum read
63 card memory is full
64 card expiration date is passed
65 card is not active (disabled)
66 card needs supervisor card

Output Parameters

None

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
#define ERROR_CARD_READ  52
WORD IsCommOK;
BYTE Result,CardType;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

IsCommOK = TAU_CardReset(&Result,&CardType);
if (IsCommOK && Result == 0) //comm ok, no card is present
{
   TAU_LcdError(ERROR_CARD_READ);
   printf("Error %d: can not read card\n",ERROR_CARD_READ);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_SetIcon

```
WORD TAU_SetIcon(BYTE Icon, BYTE IsActive);
```

Description

Turn one of the icons on the display (LCD) on or off.

Input Parameters

`BYTE Icon`

The icon number 0 - 11.

Known values:

0	**IN**	Employee entry mode
1	**BAT**	Battery low
2	**SET**	Set mode: time, parameters etc.
3	**F1**	Not in use
4	**M**	Master unit. Transactions are saved.
5	**B**	Backup unit: Transactions are overwritten.
6	**OUT**	Employee exit mode
7	**MEM**	POWERTIME unit memory is full.
8	**CARD**	Card is present (inserted).
9	**FULL**	Card is full.
10	**F2**	Not is use.
11	**NEXT**	Must insert next accumulator card

`BYTE IsActive`

Set icon on or off
0 Power off the icon
1 Power on the icon

Output Parameters

None

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
#define ICON_F1  3
WORD IsCommOK;
BYTE Result,CardType;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

IsCommOK = TAU_CardReset(&Result,&CardType);
if (IsCommOK && Result == 1) //comm ok, card is pres
{
   TAU_SetIcon(ICON_F1,TRUE);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_LcdClear

```
WORD TAU_LcdClear(void);
```

Description
Clear the display (LCD). This command does not deactivate the icons.

Input Parameters
None

Output Parameters
None

Returns
`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
#define ERROR_CARD_READ  52
WORD IsCommOK;
BYTE Result,CardType;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

IsCommOK = TAU_CardReset(&Result,&CardType);
if (IsCommOK && Result == 0) //comm ok, no card is present
{
   TAU_LcdError(ERROR_CARD_READ);
   printf("Error %d: can not read card\n",ERROR_CARD_READ);
}

//end application
TAU_LcdClear();  //clear display before exit
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_LcdTime

```
WORD TAU_LcdTime(void);
```

Description
Displaye the current time of the POWERTIME unit on the LCD.

Input Parameters
None

Output Parameters
None

Returns
`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;

TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
TAU_LcdTime();
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_LcdWriteDigit

```
WORD TAU_LcdWriteDigit(BYTE Index, BYTE Digit);
```

Description

Display a digit (ASCII character) in display position Index.

Input Parameters

`BYTE Index` — The digit position on the display: 0 - 9.

`BYTE Digit` — An ASCII character.
For example:
Digit '0': 48 (0x30)
Digit '1': 49 (0x31)
Digit '2': 50 (0x32)
Digit '3': 51 (0x33)
Digit '4': 52 (0x34)
Digit '5': 53 (0x35)
Digit '6': 54 (0x36)
Digit '7': 55 (0x37)
Digit '8': 56 (0x38)
Digit '9': 57 (0x39)
Letter 'A': 65 (0x41) etc.

Output Parameters

None

Returns

`WORD IsCommOK` — Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE n;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
```

```
//display string "0123456789"
for (n=0 ; n < 9 ; n++)
  TAU_LcdWriteDigit(n,'0'+n);

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_LcdWrite

```
WORD TAU_LcdWrite(const BYTE * str);
```

Description
Display a string on the LCD.

Input Parameters
`const BYTE * str`	ASCII String of length of 10 characters.

Output Parameters
None

Returns
`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
WORD IsCommOK;
BYTE n;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//display string "0123456789"
for (n=0 ; n < 9 ; n++)
  TAU_LcdWriteDigit(n,'0'+n);

//display string "ABCDEFGHIJ"
TAU_LcdWrite("ABCDEFGHIJ");

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_MemWrite

```
WORD TAU_MemWrite(DWORD Offset,
                  BYTE Length,
                  const BYTE * Buffer,
                  BYTE * Result);
```

Description

Writes data to the POWERTIME unit memory from a given offset on, and of a given length.

Input Parameters

`DWORD Offset`	Offset address: 0 - 32767 (32K bytes)
`BYTE Length`	Length to read: 1 - 20 (length value is limited due to the protocol).
`BYTE * Buffer`	The input buffer to write

Output Parameters

`BYTE * Result`	The output result: 0 Failed in writing memory location (offset) 1 Write OK

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Length, Result;
WORD IsCommOK, Offset;
BYTE Buffer[] = “ABCDEFGHIJ”;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf(“Communication error ...\n”);
   return;
}

Offset = 1000;
Length = sizeof(Buffer);
IsCommOK = TAU_MemWrite(Offset,Length,Buffer,&Result);

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_MemRead

```
WORD TAU_MemRead(DWORD Offset,
                 BYTE Length,
                 BYTE * Result,
                 BYTE * Buffer);
```

Description

Reads data from the POWERTIME unit memory from a given offset on, and of a given length.

Input Parameters

`DWORD Offset`	Ofsset address: 0 – 32767 (32K bytes)
`BYTE Length`	Length to read: 1 - 20 (length value is limited due to the protocol).

Output Parameters

`BYTE * Result`	The output result: 0 Can not read memory (invalid offset ?) 1 Read OK
`BYTE * Buffer`	The output buffer

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Result, Length;
WORD Offset;
BYTE Buffer[100];

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

Offset = 1000; //First transaction offset
Length = 16; //Transaction size
TAU_MemRead(Offset,Length,&Result,Buffer);

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_GetDateTime

```
WORD TAU_GetDateTime(BYTE * Result,
                     WORD * Year,
                     BYTE * Month,
                     BYTE * Day,
                     BYTE * Hour,
                     BYTE * Minute,
                     BYTE * Second);
```

Description

Reads the current date and time from the POWERTIME unit real time clock.

Input Parameters

None

Output Parameters

`BYTE * Result`	The output result: 0 Can not read date and time 1 Read OK
`WORD * Year`	The year 1990 - ...
`BYTE * Month`	The month 01 - 12.
`BYTE * Day`	The day 1 - 31
`BYTE * Hour`	The hour 0 - 23
`BYTE * Minute`	The minute 0 - 59
`BYTE * Second`	The second 0 - 59

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Result;
BYTE Month,Day,Hour,Minute,Second;
WORD Year;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
```

```
TAU_GetDateTime(&Result,&Year,&Month,&Day,&Hour,&Minute,&Second);

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_SetDateTime

```
WORD TAU_SetDateTime(WORD Year,
                     BYTE Month,
                     BYTE Day,
                     BYTE Hour,
                     BYTE Minute,
                     BYTE Second
                     BYTE * Result);
```

Description

Write a given date and time to the POWERTIME unit real time clock.

Input Parameters

`WORD Year`	The year 1990 - ...
`BYTE Month`	The month 01 - 12.
`BYTE Day`	The day 1 - 31
`BYTE Hour`	The hour 0 - 23
`BYTE Minute`	The minute 0 - 59
`BYTE Second`	The second 0 - 59

Output Parameters

`BYTE * Result`

The output result:

0 Can not write date and time

1 Write OK

Returns

`WORD IsCommOK`

Communication status:

0 Communication error

1 Communication OK

Example

```
BYTE Result = 0;
WORD IsCommOK;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
```

```
TAU_SetDateTime(1999,11,20,23,59,59,&Result); //20/11/1999 23:59:59
if (Result == 0) //error
  TAU_LcdError(1); //display error 001 (unkonwn error)
else //ok
  TAU_LcdTime(); //display time

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_GetTotalTransactions

```
WORD TAU_GetTotalTransactions(BYTE * Result,
                              WORD * Total);
```

Description

Read the total number of transactions stored in the TAU.

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 Memory read error
1 Memory read OK

`WORD * Total`

Total number of transactions stored in the TAU:
0 - 1800 (Stored transaction size is 16 bytes)

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber;
DWORD SupervisorNumber;
WORD Year;
WORD CardEnum, UnitEnum;
BYTE Month, Day, Hour, Minute, Event;


//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//download transactions
IsCommOK = TAU_GetTo(&Result,&nTotal);
```

```
if (IsCommOK)
{
   //"download" transactions
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAU_ReadTransaction(nIndex,
                          &Result,
                          &Year,&Month,&Day,&Hour,&Minute,
                          &CardNumber,&CardEnum,&UnitEnum,
                          &SupervisorNumber,
                          &Event);
      //print transactions date & time
      printf("Transaction no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,Day,Month,Year,Hour,Minute);

   }

   printf("Total transactions downloaded: %d\n",nTotal);

   //erase transaction from TAU after reading all transactions
   TAU_EraseTransactions(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_ReadTransaction

```
WORD TAU_ReadTransaction(WORD Index,
                         BYTE * Result,
                         WORD * Year,
                         BYTE * Month,
                         BYTE * Day,
                         BYTE * Hour,
                         BYTE * Minute,
                         DWORD * CardNumber,
                         WORD * CardEnumerator,
                         WORD * UnitEnumerator,
                         DWORD * Supervisor,
                         BYTE * Event);
```

Description

Read a transaction from the TAU.

Input Parameters

`WORD Index`	A transaction record number: 0 - Total Total is the value receive by calling function `TAU_GetTotalTransactions`

Output Parameters

`BYTE * Result`	The output result: 0 Memory read error 1 Memory read OK
`WORD * Year`	The transaction year 1990 - ...
`BYTE * Month`	The transaction month 01 - 12.
`BYTE * Day`	The transaction day 1 - 31
`BYTE * Hour`	The transaction hour 0 – 23
`BYTE * Minute`	The transaction minute 0 – 59
`DWORD * CardNumber`	The employee card number: 1 – 999,999,999
`WORD * CardEnum`	The employee card transaction enumerator: 0 - 9,999
`WORD * UnitEnum`	The POWERTIME unit transaction enumerator: 0 - 9,999
`DWORD * Supervisor`	The supervisor card number: 1 - 999,999,999 If value is equal to 0, a supervisor card was not required.
`BYTE * Event`	The employee action: 0 – Entry (in) 1 - Exit (out)

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber;
DWORD SupervisorNumber;
WORD Year;
WORD CardEnum, UnitEnum;
BYTE Month, Day, Hour, Minute, Event;


//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//download transactions
IsCommOK = TAU_GetTotalTransactions(&Result,&nTotal);

if (IsCommOK)
{
   //"download" transactions
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAU_ReadTransaction(nIndex,
                          &Result,
                          &Year,&Month,&Day,&Hour,&Minute,
                          &CardNumber,&CardEnum,&UnitEnum,
                          &SupervisorNumber,
                          &Event);
      //print transactions date & time
      printf("Transaction no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,Day,Month,Year,Hour,Minute);

   }

   printf("Total transactions downloaded: %d\n",nTotal);

   //erase transaction from TAU after reading all transactions
   TAU_EraseTransactions(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_EraseTransactions

```
WORD TAU_EraseTransactions(BYTE * Result);
```

Description

Erase all transactions stored in the TAU.

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 Memory write error
1 Memory write OK

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nTotal = 0;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
//clear transactions from TAU
IsCommOK = TAU_GetTotalTransactions(&Result,&nTotal);
if (IsCommOK && nTotal > 0)
{
   //erase transaction from TAU
   TAU_EraseTransactions(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_GetTotalActivities

```
WORD TAU_GetTotalActivities(BYTE * Result,
                            WORD * Total);
```

Description

Read the total number of supervisor activities stored in the TAU.

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 Memory read error
1 Memory read OK

`WORD * Total`

Total number of activities stored in the TAU:
0 - 100 (Activity size stored is 16 bytes)

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD SupervisorNumber;
WORD YearOld, YearNew;
WORD CardEnum, UnitEnum;
BYTE MonthOld, DayOld, HourOld, MinuteOld;
BYTE MonthNew, DayNew, HourNew, MinuteNew;


//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//download activities
IsCommOK = TAU_GetTotalActivities(&Result,&nTotal);
```

```
if (IsCommOK)
{
   //"download" activities
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAU_ReadActivity(nIndex,
                       &Result,
                       &SupervisorNumber,
                       &YearOld,&MonthOld,&DayOld,
                       &HourOld,&MinuteOld,
                       &YearNew,&MonthNew,&DayNew,
                       &HourNew,&MinuteNew,
                       &CardEnum,&UnitEnum);
      //print activity old date & time
      printf("Activity no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,DayOld,MonthOld,YearOld,HourOld,MinuteOld);

   }

   printf("Total activities downloaded: %d\n",nTotal);

   //erase activities from TAU after reading all
   TAU_EraseActivities(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_ReadActivity

```
WORD TAU_ReadActivity(WORD Index,
                      BYTE * Result,
                      DWORD * Supervisor,
                      WORD * YearOld,
                      BYTE * MonthOld,
                      BYTE * DayOld,
                      BYTE * HourOld,
                      BYTE * MinuteOld,
                      WORD * YearNew,
                      BYTE * MonthNew,
                      BYTE * DayNew,
                      BYTE * HourNew,
                      BYTE * MinuteNew,
                      WORD * CardEnumerator,
                      WORD * UnitEnumerator);
```

Description

Read an activity record from the TAU.

Input Parameters

`WORD Index`	An activity record number: 0 - Total Total is the value received by calling function `TAU_GetTotalActivities`

Output Parameters

`BYTE * Result`	The output result: 0 Memory read error 1 Memory read OK
`DWORD * Supervisor`	The supervisor card number: 1 - 999,999,999
`WORD * YearOld`	The previous year 1990 - ...
`BYTE * MonthOld`	The previous month 01 - 12.
`BYTE * DayOld`	The previous day 1 - 31
`BYTE * HourOld`	The previous hour 0 – 23
`BYTE * MinuteOld`	The previous minute 0 – 59
`WORD * YearNew`	The new year 1990 - ...
`BYTE * MonthNew`	The new month 01 - 12.
`BYTE * DayNew`	The new day 1 - 31
`BYTE * HourNew`	The new hour 0 - 23
`BYTE * MinuteNew`	The new minute 0 - 59
`WORD * CardEnum`	The Supervisor card transactions enumerator: 0 - 9,999

`WORD * UnitEnum`	The POWERTIME unit transactions enumerator: 0 - 9,999

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD SupervisorNumber;
WORD YearOld, YearNew;
WORD CardEnum, UnitEnum;
BYTE MonthOld, DayOld, HourOld, MinuteOld;
BYTE MonthNew, DayNew, HourNew, MinuteNew;


//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//download activities
IsCommOK = TAU_GetTotalActivities(&Result,&nTotal);
if (IsCommOK)
{
   //"download" activities
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAU_ReadActivity(nIn,
                       &Result,
                       &SupervisorNumber,
                       &YearOld,&MonthOld,&DayOld,
                       &HourOld,&MinuteOld,
                       &YearNew,&MonthNew,&DayNew,
                       &HourNew,&MinuteNew,
                       &CardEnum,&UnitEnum);
      //print activity old date & time
      printf("Activity no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,DayOld,MonthOld,YearOld,HourOld,MinuteOld);

   }

   printf("Total activities downloaded: %d\n",nTotal);
```

```
   //erase activities from TAU after reading all
   TAU_EraseActivities(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_EraseActivities

```
WORD TAU_EraseActivities(BYTE * Result);
```

Description
Erase all activities stored in the TAU.

Input Parameters
None.

Output Parameters
`BYTE * Result`

The output result:
0 Memory write error
1 Memory write OK

Returns
`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nTotal = 0;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//clear activities from TAU
IsCommOK = TAU_GetTotalActivities(&Result,&nTotal);
if (IsCommOK && nTotal > 0)
{
   //erase transaction from TAU
   TAU_EraseActivities(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_ResetParameters

```
WORD TAU_ResetParameters(void);
```

Description

Reset the POWERTIME unit parameters (e.g. unit number, total transactions, flags etc.).
Default values are set. This command can be executed for new units, powered up for the first time after production, or to renew the unit.

Input Parameters

None.

Output Parameters

None.

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
BYTE IsDebug = TRUE;
BYTE Result;

//begin application
TAU_CommOpen(IsDebug);
TAU_CommConnect();

//initialize the TAU
TAU_ResetParameters();
TAU_EraseTransactions();
TAU_EraseActivities();
TAU_SetDateTime(1999,11,20,23,59,59,&Result); //20/11/1999 23:59:59
TAU_Beep(400);

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_ReadParameter

```
WORD TAU_ReadParameter(BYTE Offset,
                       BYTE Length,
                       BYTE * Result,
                       DWORD * Value);
```

Description

Reads one parameter from the POWERTIME unit memory.

Input Parameters

`BYTE Offset`	Ofsset address: 0 - 49 (50 bytes only)
`BYTE Length`	Length to read: 1 - 4 (bytes)

Output Parameters

`BYTE * Result`	The output result: 0 Can not read memory (invalid offset ?) 1 Read OK
`DWORD * Value`	The output parameter value 0 - 999,999,999 (depends on parameter length)

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Result, Length;
BYTE Offset;
DWORD Value = 0;
WORD IsCommOK;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//read unit ID
Offset = 2;
Length = 3;
```

```
TAU_ReadParameter(Offset,Length,&Result,&Value);
printf("Unit ID is %ld .\n",Value);

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

Known Parameters

Non-volatile Parameters

Written once when initializing the POWERTIME unit for the first time. The user application can read them, but it is recommended never to modify them.

Offset	Length	Value Range	Name
2	3	1 - 9,999,999	Unit ID

Volatile Parameters

Written when initializing the POWERTIME unit for the first time, and can be read and written whenever needed.

Offset	Length	Value Range	Name
5	1	0 - 1	Is Backup Unit
30	1	0 - 99	Supervisor Permission Timeout (minutes)
34	1	0 - 1	Is Check Cards Expiration Date
35	1	0 - 1	Is Rewrite Accumulator Cards (when full)
36	1	0 - 1	Is Supervisor Card is needed for all employees
37	1	0 - 1	Is Online (communication)
38	1	0 - 99	Transactions Notification Percent (indicate full)

Run Time Read-Only Parameters

The user application can read them, but it is recommended that they not be modified.

Offset	Length	Value Range	Name
6	2	0 - 9,999	Total Transactions
8	2	0 - 9,999	Current Transaction
10	2	0 - 9,999	Total Activities
12	2	0 - 9,999	Current Activity
14	2	0 - 9,999	Total Transaction Old
16	2	0 - 9,999	Total Activity Old
18	2	1 - 9,999	Transactions Enumerator (unit enumerator)
20	2	1 - 9,999	Activities Enumerator (unit enumerator)
22	1	1 - 4	Last Card Type: 1 = Personal card (employee) 2 = Accumulator card 3 = Supervisor card 4 = Setup (unit) card
23	4	0 - 999,999,999	Last Supervisor Card Number
27	2	0 - 9,999	Discharge Counter
29	1	0 - 1	In / Out Status (entry or exit)
30	1	0 - 99	Supervisor Permission Timeout (minutes)
39	2	0 - 1,800	First transaction to write to accumulator (next)
41	2	0 - 100	First activity to write to accumulator (next)

Other Parameters

Offset	Length	Value Range	Name
0	1	0	Reserved 1
1	1	0	Reserved 2

TAU_WriteParameter

```
WORD TAU_WriteParameter(BYTE Offset,
                        BYTE Length,
                        DWORD Value,
                        BYTE * Result);
```

Description

Write one parameter to the POWERTIME unit memory.

Input Parameters

`BYTE Offset`	Ofsset address: 0 - 49 (50 bytes only)
`BYTE Length`	Length to read: 1 - 4 (bytes)
`DWORD Value`	The input parameter value 0 - 999,999,999 (depend on parameter length)

Output Parameters

`BYTE * Result`	The output result: 0 Can not write memory (invalid offset ?) 1 Write OK

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Result, Length;
BYTE Offset;
DWORD Value = 1234;

//begin application
TAU_CommOpen(FALSE);
TAU_CommConnect();

//write unit ID
Offset = 2;
Length = 3;
TAU_WriteParameter(Offset,Length,,Value,&Result);
printf("Written unit ID is %ld .\n",Value);

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_UserDataRead

```
WORD TAU_UserDataRead(DWORD Offset,
                      BYTE Length,
                      BYTE * Result,
                      BYTE * Buffer);
```

Description

Reads user data in the POWERTIME unit memory from a given offset, of a given length.
The user application can write unique data to the TAU, and read it later. This data can be up to 256 bytes long.

Input Parameters

`DWORD Offset`	Offset address: 0 - 255
`BYTE Length`	Length to read: 1 - 20 (length value is limited due to the protocol).

Output Parameters

`BYTE * Result`	The output result: 0 Can not read memory (invalid offset ?) 1 Read OK
`BYTE * Buffer`	The output buffer

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Result, Length;
WORD Offset;
BYTE OwnerName[] = "MyName";
BYTE TauOwnerName[20];

//begin application
TAU_CommOpen(FALSE);
TAU_CommConnect();

Offset = 0;
Length = sizeof(OwnerName);
TAU_UserDataRead(Offset,Length,&Result,TauOwnerName);
if (strcmp(OwnerName,TauOwnerName)!= 0) //not equal
{
   printf("This TAU unit is not owned by %s !!! \n",OwnerName);
}
```

```
//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAU_UserDataWrite

```
WORD TAU_UserDataWrite(DWORD Offset,
                       BYTE Length,
                       const BYTE * Buffer,
                       BYTE * Result);
```

Description

Write user data to the POWERTIME unit memory from a given offset, of a given length.
The user application can write unique data to the TAU, and read it later.

Input Parameters

`DWORD Offset`	Ofsset address: 0 – 255
`BYTE Length`	Length to read: 1 - 20 (length value is limited due to the protocol).
`BYTE * Buffer`	The input buffer to write

Output Parameters

`BYTE * Result`	The output result: 0 Failed in writing memory location (offset) 1 Write OK

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
BYTE Result, Length;
WORD Offset;
BYTE OwnerName[] = “MyName”;

//begin application
TAU_CommOpen(FALSE);
TAU_CommConnect();

Offset = 0;
Length = sizeof(OwnerName);
TAU_UserDataWrite(Offset,Length,OwnerName,&Result);
if (Result != 0)
  printf(“This TAU unit is now owned by %s !!!\n”,OwnerName);

//end application
TAU_CommClose();
```

TAC_CardReset

```
WORD TAC_CardReset(BYTE * Result,
                   BYTE * ChipType,
                   BYTE * CardType);
```

Description

Resets the card, reads it (all except transactions) and returns the type of the chip on card and the card type (supervisor, employee, accumulator or setup).

Notes:

- **This command takes a long time (about 3 seconds) !!!**
- **This command must come before any transaction or activity reading (or total query).**

Input Parameters

None.

Output Parameters

`BYTE * Result`	The output result: 0 Checksum error on card, or card can not be read. 1 read OK.
`BYTE * ChipType`	The chip type (known values): 00 Unknown chip type or card is not inserted 34 (equal to 22H in hex notation) -- 2K byte memory card 40 (equal to 28H in hex notation) -- 8K byte memory card
`BYTE * CardType`	The card type (known values): 1 Employee card (personal card) 2 Accumulator card 3 Supervisor card 4 Setup card (unit card)

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
WORD IsCommOK, nTotal, nIndex;
BYTE Result;
BYTE ChipType;
BYTE CardType;
```

```
//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
IsCommOK = TAC_CardReset(&Result,&ChipType,&CardType);
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
if (Result == 1 && ChipType != 0)
{
    TAC_GetTotalTransactions(&Result,&nTotal);
    for (nIndex=0 ; nIndex < nTotal ; nIndex++)
    {
        TAC_ReadTransaction(nIndex,&Result,...);  //...data
        //do something with transaction
    }
    //end of "download"
    TAC_EraseTransactions(&Result);  //...data
}
//end application
TAU_CommDisconnect();
TAU_CommClose();
```

Known Card Types

Card Type	Name	Chip Type	Transactions	Activities
1	Employee Card (personal)	2K	Yes	No
2	Accumulator Card	8K	Yes	Yes
3	Supervisor Card	2K	No	Yes
4	Setup Card (unit)	2K	No	No

Transactions/Activities Actions Order

1) Reset the card (TAC_CardReset)
2) Get total transactions
3) Read transaction from 0 to (total-1)
4) Erase transaction (if needed ...)

TAC_GetTotalTransactions

```
WORD TAC_GetTotalTransactions(BYTE * Result,
                              WORD * Total);
```

Description

Read the total number of transactions stored in the POWERTIME card (memory card).

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 Card read error
1 Card read OK

`WORD * Total`

Total transactions stored in the card:
For 2K card:
0 - 121 (Size of stored transaction is 16 bytes)
For 8K card:
0 - 505 (Size of stored transaction is 16 bytes)

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber,UnitNumber;
DWORD SupervisorNumber;
WORD Year;
WORD CardEnum, UnitEnum;
BYTE Month, Day, Hour, Minute, Event;


//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
```

```
//must reset the card !!!
IsCommOK = TAC_CardReset(&Result,&ChipType,&CardType);
if (!IsCommOK || ChipType ==0)
{
   printf("Can not read card...\n");
   return;
}
//download transactions
IsCommOK = TAC_GetTotalTransactions(&Result,&nTotal);
if (IsCommOK)
{
   //"download" transactions
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAC_ReadTransaction(nIndex,
                          &Result,
                          &Year,&Month,&Day,&Hour,&Minute,
                          &CardNumber,&UnitNumber,
                          &CardEnum,&UnitEnum,
                          &SupervisorNumber,
                          &Event);
      //print transactions date & time
      printf("Transaction no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,Day,Month,Year,Hour,Minute);

   }

   printf("Total transactions downloaded: %d\n",nTotal);

   //erase transaction from TAU after reading all transactions
   TAC_EraseTransactions(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAC_ReadTransaction

```
WORD TAC_ReadTransaction(WORD Index,
                         BYTE * Result,
                         WORD * Year,
                         BYTE * Month,
                         BYTE * Day,
                         BYTE * Hour,
                         BYTE * Minute,
                         DWORD * CardNumber,
                         DWORD * UnitNumber,
                         WORD * CardEnumerator,
                         WORD * UnitEnumerator,
                         DWORD * Supervisor,
                         BYTE * Event);
```

Description

Read a transaction from the POWERTIME card (memory card).

Input Parameters

`WORD Index`	A transaction record number: 0 - (Total-1) Total is the value obtained by calling function `TAC_GetTotalTransactions`

Output Parameters

`BYTE * Result`	The output result: 0 Card read error 1 Card read OK
`WORD * Year`	The transaction year 1990 - ...
`BYTE * Month`	The transaction month 01 - 12.
`BYTE * Day`	The transaction day 1 - 31
`BYTE * Hour`	The transaction hour 0 – 23
`BYTE * Minute`	The transaction minute 0 – 59
`DWORD * CardNumber`	The employee card number: 1 - 999,999,999
`DWORD * UnitNumber`	The unit number: 1 - 9,999,999
`WORD * CardEnum`	The employee card transactions enumerator: 0 - 9,999
`WORD * UnitEnum`	The POWERTIME unit transactions enumerator: 0 - 9,999
`DWORD * Supervisor`	The supervisor card number: 1 – 999,999,999 If value equal to 0, no supervisor card was needed
`BYTE * Event`	The employee action: 0 - Entry (in) 1 - Exit (out)

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber,UnitNumber;
DWORD SupervisorNumber;
WORD Year;
WORD CardEnum, UnitEnum;
BYTE Month, Day, Hour, Minute, Event;


//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
//must reset the card !!!
IsCommOK = TAC_CardReset(&Result,&ChipType,&CardType);
if (!IsCommOK || ChipType ==0)
{
   printf("Can not read card...\n");
   return;
}
//download transactions
IsCommOK = TAC_GetTotalTransactions(&Result,&nTotal);
if (IsCommOK)
{
   //"download" transactions
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAC_ReadTransaction(nIndex,
                          &Result,
                          &Year,&Month,&Day,&Hour,&Minute,
                          &CardNumber,&UnitNumber,
                          &CardEnum,&UnitEnum,
                          &SupervisorNumber,
                          &Event);
      //print transactions date & time
      printf("Transaction no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,Day,Month,Year,Hour,Minute);

   }
```

```
   printf("Total transactions downloaded: %d\n",nTotal);

   //erase transaction from TAU after reading all transactions
   TAC_EraseTransactions(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAC_EraseTransactions

```
WORD TAC_EraseTransactions(BYTE * Result);
```

Description

Erase all transactions stored in the POWERTIME card (memory card).

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 Card write error
1 Card write OK

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber,UnitNumber;
DWORD SupervisorNumber;
WORD Year;
WORD CardEnum, UnitEnum;
BYTE Month, Day, Hour, Minute, Event;


//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
//must reset the card !!!
IsCommOK = TAC_CardReset(&Result,&ChipType,&CardType);
if (!IsCommOK || ChipType ==0)
{
   printf("Can not read card...\n");
   return;
}
```

```
//download transactions
IsCommOK = TAC_GetTotalTransactions(&Result,&nTotal);
if (IsCommOK)
{
   //"download" transactions
   for (nIndex=0 ; nInd< nTotal ; nIndex++)
   {
      TAC_ReadTransaction(nIndex,
                          &Result,
                          &Year,&Month,&Day,&Hour,&Minute,
                          &CardNumber,&UnitNumber,
                          &CardEnum,&UnitEnum,
                          &SupervisorNumber,
                          &Event);
      //print transactions date & time
      printf("Transaction no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,Day,Month,Year,Hour,Minute);

   }

   printf("Total transactions downloaded: %d\n",nTotal);

   //erase transaction from TAU after reading all transactions
   TAC_EraseTransactions(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAC_GetTotalActivities

```
WORD TAC_GetTotalActivities(BYTE * Result,
                            WORD * Total);
```

Description

Read the total number of supervisor activities stored in the POWERTIME card.

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 Card read error
1 Card read OK

`WORD * Total`

Total activities stored in the card:
For 2K card:
0 - 121 (Size of stored transaction is 16 bytes)
For 8K card:
0 - 505 (Size of stored transaction is 16 bytes)

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber,UnitNumber;
WORD YearOld, YearNew;
WORD CardEnum, UnitEnum;
BYTE MonthOld, DayOld, HourOld, MinuteOld;
BYTE MonthNew, DayNew, HourNew, MinuteNew;
BYTE CardType,ChipType;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
```

```
//must reset the card !!!
IsCommOK = TAC_CardReset(&Result,&ChipType,&CardType);
if (!IsCommOK || ChipType ==0)
{
   printf("Can not read card...\n");
   return;
}

//download activities
IsCommOK = TAC_GetTotalActivities(&Result,&nTotal);
if (IsCommOK)
{
   //"download" activities
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAC_ReadActivity(nIndex,
                       &Result,
                       &CardNumber,&UnitNumber,
                       &YearOld,&MonthOld,&DayOld,
                       &HourOld,&MinuteOld,
                       &YearNew,&MonthNew,&DayNew,
                       &HourNew,&MinuteNew,
                       &CardEnum,&UnitEnum);
      //print activity old date & time
      printf("Activity no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,DayOld,MonthOld,YearOld,HourOld,MinuteOld);

   }

   printf("Total activities downloaded: %d\n",nTotal);

   //erase activities from TAU after reading all
   TAC_EraseActivities(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAC_ReadActivity

```
WORD TAC_ReadActivity(WORD Index,
                      BYTE * Result,
                      DWORD * CardNumber,
                      DWORD * UnitNumber,
                      WORD * YearOld,
                      BYTE * MonthOld,
                      BYTE * DayOld,
                      BYTE * HourOld,
                      BYTE * MinuteOld,
                      WORD * YearNew,
                      BYTE * MonthNew,
                      BYTE * DayNew,
                      BYTE * HourNew,
                      BYTE * MinuteNew,
                      WORD * CardEnumerator,
                      WORD * UnitEnumerator);
```

Description

Read a transaction from the POWERTIME card.

Input Parameters

`WORD Index`	An activity record number: 0 - (Total-1) Total is the value obtained by calling function `TAC_GetTotalActivities`

Output Parameters

`BYTE * Result`	The output result: 0 Card read error 1 Card read OK
`DWORD * CardNumber`	The supervisor card number: 1 - 999,999,999
`DWORD * UnitNumber`	The unit number: 1 - 9,999,999
`WORD * YearOld`	The old year 1990 - ...
`BYTE * MonthOld`	The old month 01 - 12.
`BYTE * DayOld`	The old day 1 - 31
`BYTE * HourOld`	The old hour 0 - 23
`BYTE * MinuteOld`	The old minute 0 - 59
`WORD * YearNew`	The new year 1990 - ...
`BYTE * MonthNew`	The new month 01 - 12.
`BYTE * DayNew`	The new day 1 - 31
`BYTE * HourNew`	The new hour 0 - 23
`BYTE * MinuteNew`	The new minute 0 - 59

`WORD * CardEnum`	The employee card transactions enumerator: 0 - 9,999
`WORD * UnitEnum`	The POWERTIME unit transactions enumerator: 0 - 9,999

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber,UnitNumber;
WORD YearOld, YearNew;
WORD CardEnum, UnitEnum;
BYTE MonthOld, DayOld, HourOld, MinuteOld;
BYTE MonthNew, DayNew, HourNew, MinuteNew;
BYTE CardType,ChipType;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}

//must reset the card !!!
IsCommOK = TAC_CardReset(&Result,&ChipType,&CardType);
if (!IsCommOK || ChipType ==0)
{
   printf("Can not read card...\n");
   return;
}

//download activities
IsCommOK = TAC_GetTotalActivities(&Result,&nTotal);
if (IsCommOK)
{
   //"download" activities
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAC_ReadActivity(nIndex,
                       &Result,
                       &CardNumber,&UnitNumber,
                       &YearOld,&MonthOld,&DayOld,
                       &HourOld,&MinuteOld,
                       &YearNew,&MonthNew,&DayNew,
                       &HourNew,&MinuteNew,
                       &CardEnum,&UnitEnum);
      //print activity old date & time
```

```
      printf("Activity no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,DayOld,MonthOld,YearOld,HourOld,MinuteOld);

   }

   printf("Total activities downloaded: %d\n",nTotal);

   //erase activities from TAU after reading all
   TAC_EraseActivities(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAC_EraseActivities

```
WORD TAC_EraseActivities(BYTE * Result);
```

Description

Erase all activities stored in the POWERTIME card (memory card).

Input Parameters

None.

Output Parameters

`BYTE * Result`

The output result:
0 Card write error
1 Card write OK

Returns

`WORD IsCommOK`

Communication status:
0 Communication error
1 Communication OK

Example

```
WORD IsCommOK;
BYTE Result;
WORD nIndex;
WORD nTotal = 0;
DWORD CardNumber,UnitNumber;
WORD YearOld, YearNew;
WORD CardEnum, UnitEnum;
BYTE MonthOld, DayOld, HourOld, MinuteOld;
BYTE MonthNew, DayNew, HourNew, MinuteNew;
BYTE CardType,ChipType;

//begin application
TAU_CommOpen(FALSE);
IsCommOK = TAU_CommConnect();
if (!IsCommOK)
{
   printf("Communication error ...\n");
   return;
}
//must reset the card !!!
IsCommOK = TAC_CardReset(&Result,&ChipType,&CardType);
if (!IsCommOK || ChipType ==0)
{
   printf("Can not read card...\n");
   return;
}
```

```
//download activities
IsCommOK = TAC_GetTotalActivities(&Result,&nTotal);
if (IsCommOK)
{
   //"download" activities
   for (nIndex=0 ; nIndex < nTotal ; nIndex++)
   {
      TAC_ReadActivity(nIndex,
                       &Result,
                       &CardNumber,&UnitNumber,
                       &YearOld,&MonthOld,&DayOld,
                       &HourOld,&MinuteOld,
                       &YearNew,&MonthNew,&DayNew,
                       &HourNew,&MinuteNew,
                       &CardEnum,&UnitEnum);
      //print activity old date & time
      printf("Activity no. %d: %02d/%02d/%04d %02d:%02d\n",
             nIndex,DayOld,MonthOld,YearOld,HourOld,MinuteOld);

   }

   printf("Total activities downloaded: %d\n",nTotal);

   //erase activities from TAU after reading all
   TAC_EraseActivities(&Result);
}

//end application
TAU_CommDisconnect();
TAU_CommClose();
```

TAC_FormatWrite

```
WORD TAC_FormatWrite(BYTE CardType,
                     DWORD CardNumber,
                     const BYTE * LastName, //20
                     const BYTE * FirstName,//20
                     WORD IssueYear,
                     BYTE IssueMonth,
                     BYTE IssueDay,
                     WORD ExpYear,
                     BYTE ExpMonth,
                     BYTE ExpDay,
                     BYTE IsActive,
                     BYTE IsSupervisor,
                     WORD IsBackup,
                     BYTE * Result)
```

Description

Create a new POWERTIME card (employee, accumulator, supervisor, setup). This creation destroys any previous data that exists on the card.

Notes:

- **This command does not require prior resetting of the POWERTIME card (*TAC_CardReset*).**
- **This command may take few seconds.**

Input Parameters

`BYTE CardType`	The card type (known values): 1 Employee card (personal card) 2 Accumulator card 3 Supervisor card 4 Setup card (unit card)
`DWORD CardNumber`	The card number: 1 - 999,999,999
`BYTE * LastName`	The Employee/Supervisor/Card Owner last name: 20 characters. Name terminates with zero.
`BYTE * FirstName`	The Employee/Supervisor/Card Owner first name: 20 characters. Name terminates with zero.
`WORD IssueYear`	Creation year: 1990 -
`BYTE IssueMonth`	Creation Month: 1 - 12
`BYTE IssueDay`	Creation Day: 1 - 31

`WORD ExpYear`	Expiration Year: 1990 -
`BYTE ExpMonth`	Expiration Month: 1 - 12
`BYTE ExpDay`	Expiration Day: 1 - 31
`WORD IsActive`	Is active card (for all card types): 0 Card is not active (normally not used) 1 Card is active (should be used)
`BYTE IsSupervisor`	Is Supervisor confirmation required (for Employee card): Does the Employee card need the confirmation of a supervisor in order to create an In or Out transaction in the TAU? 0 = No 1 = Yes
`BYTE IsBackup`	Is this a Backup card (Employee & Supervisor cards): In a backup card, transactions are stored in a cyclic manner. The primary storage of employee transactions and supervisor activities is in the POWERTIME unit. 0 = not a backup card (normal) 1 = a backup card

Output Parameters

`BYTE * Result`	The output result: 0 Card write error 1 Card write OK

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

N/A

TAC_FormatRead

```
WORD TAC_FormatRead(BYTE * Result,
                    BYTE * CardType,
                    DWORD * CardNumber,
                    BYTE * LastName, //20
                    BYTE * FirstName,//20
                    WORD * IssueYear,
                    BYTE * IssueMonth,
                    BYTE * IssueDay,
                    WORD * ExpYear,
                    BYTE * ExpMonth,
                    BYTE * ExpDay,
                    BYTE * IsActive,
                    BYTE * IsSupervisor,
                    WORD * IsBackup)
```

Description

Reads the formatted (constant) data from the TAC card (employee, accumulator, supervisor, setup) .

Notes:

- **This command does not require prior resetting of the POWERTIME card (*TAC_CardReset*).**
- **This command may take few seconds.**

Input Parameters

None

Output Parameters

`BYTE * Result`	The output result: 0 Card read error (or check sum error) 1 Card read OK (and check sum OK)
`BYTE * CardType`	The card type (known values): 1 Employee card (personal card) 2 Accumulator card 3 Supervisor card 4 Setup card (unit card)
`DWORD * CardNumber`	The card number: 1 - 999,999,999
`BYTE * LastName`	The Employee/Supervisor/Card Owner last name: 20 characters. Name terminates with zero.

`BYTE * FirstName`	The Employee/Supervisor/Card Owner first name: 20 characters. Name terminates with zero.
`WORD * IssueYear`	Creation year: 1990 -
`BYTE * IssueMonth`	Creation Month: 1 – 12
`BYTE * IssueDay`	Creation Day: 1 – 31
`WORD * ExpYear`	Expiration Year: 1990 -
`BYTE * ExpMonth`	Expiration Month: 1 – 12
`BYTE * ExpDay`	Expiration Day: 1 – 31
`WORD * IsActive`	Is active card (for all card types): 0 Card is not active (normally not used) 1 Card is active (should be used)
`BYTE * IsSupervisor`	Is Supervisor confirmation required (for Employee card): Does the Employee card need the confirmation of a supervisor in order to create an In or Out transaction in the TAU? 0 = No 1 = Yes
`BYTE * IsBackup`	Is this a Backup card (Employee & Supervisor cards): In backup card transactions are stored in a cyclic manner. The employee transactions or supervisor activities are stored in the POWERTIME unit. 0 = not a backup card (normal) 1 = a backup card

Returns

`WORD IsCommOK`	Communication status: 0 Communication error 1 Communication OK

Example

N/A