TrafficMonitor®

ТМ API - ИНТЕРФЕЙС ПРИКЛАДНОГО ПРОГРАММИРОВАНИЯ ДЛЯ УПРАВЛЕНИЯ УСТРОЙСТВАМИ Трафик-Монитор®

ТМ API
Справочное руководство

Версия 3.0

ЮФКВ.20094-01 96 01-001ФЛ К

НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

Содержание

Список иллюстраций

1. Используемые сокращения

- ТМ - **Трафик-Монитор®** - устройство детектирования транспортных средств и мониторинга дорожного движения.
- ТМСР - **TrafficMonitor®** Control Protocol - протокол управления устройством **Трафик-Монитор®**.
- ТМ API - интерфейс прикладного программирования для управления устройством **Трафик-Монитор®** по протоколу ТМСР.
- ТМСРС - программный модуль (библиотека динамической загрузки), реализующий интерфейс прикладного программирования.
- ТС - транспортное средство.
- УС - Управляющая станция. Персональный компьютер, который подключается к ТМ, управляет его работой и получает от него данные.

2. Общие сведения

Программный модуль TMCPC работает в среде Windows (2000/XP)/Linux, является библиотекой динамической загрузки с C/C++ интерфейсом и служит для создания программ для удалённого взаимодействия с приложением **Трафик-Монитор®**. TMCPC предоставляет хэндл-ориентированный интерфейс для настройки TM, переключения режимов работы и получения данных.

3. Подключаемые модули и заголовочные файлы

Для работы с программным модулем необходимо включить модуль TMCPCl в область "видимости" операционной системы. Одно из решений - создание переменной среды окружения, например, с именем TMCPC, в которой записывается путь к каталогу с заголовочными библиотечными и бинарными файлами TMCPC, и добавление созданной переменной к переменной окружения PATH.

Файлы для разработки программ с использованием TMCPC:

- TMCPC.dll/TMCPC.so - программный модуль
- TMCPC.lib - модуль для раннего связывания программ с TMCPC в среде MSVC++.
- TMCPC.h - заголовочный файл с описанием структур и функций API
- TMCP.h - заголовочный файл с описанием структур и идентификаторов протокола TMCP

Для использования библиотеки в C/C++ программах включите в исходный файл директиву #include "TMCPC.h". Если используется среда разработки MSVC++, то для раннего связывания подключите к проекту файл TMCPC.lib. Используйте переменную окружения TMCPC для задания пути к файлам TMCPC.h и TMCPC.lib.

4. Описание типов, идентификаторов, структур и функций TMCPC

4.1. Типы данных

Тип	Тип C/C++	Размер (байт)
TMCPC_HANDLE	int32_t	4

Объявлен в TMCPC.h.

4.2. Перечисления

4.2.1. TMCP_COMMAND

Команды

```
  typedef enum tagTMCP_COMMAND {
    TMCP_COMM_AUTHENTICATION = 128,
    TMCP_COMM_CHANGE_PASSWORD,
    TMCP_COMM_CLEAR_DATA,
    TMCP_COMM_CLIENT_SERVER_SETTINGS,
    TMCP_COMM_START,
    TMCP_COMM_STOP,
    TMCP_COMM_AFFIXMENT,
    TMCP_COMM_LANES,
    TMCP_COMM_STAT_NOTIF,
    TMCP_COMM_ACC_PERIOD,
    TMCP_COMM_IP_CAM_URI
} TMCP_COMMAND;
```

- *TMCP_COMM_AUTHENTICATION* - аутентификация.
- *TMCP_COMM_CHANGE_PASSWORD* - смена пароля администратора.
- *TMCP_COMM_CLEAR_DATA* - удаление накопленных данных.
- *TMCP_COMM_CLIENT_SERVER_SETTINGS* - настройки клиент-сервера.
- *TMCP_COMM_START* - запуск TM.
- *TMCP_COMM_STOP* - останов TM.

- *TMCP_COMM_AFFIXMENT* - установка привязки.
- *TMCP_COMM_LANES* - установка параметров полос движения.
- *TMCP_COMM_STAT_NOTIF* - настройка уведомления о статистике.
- *TMCP_COMM_ACC_PERIOD* - установка периода накопления текущей статистики.
- *TMCP_COMM_IP_CAM_URI* - установка URI IP камеры.

Объявлено в TMCP.h

4.2.2. TMCP_FIND_DIRECTION

Направление поиска при выборки записи из БД

```
  typedef enum tagTMCP_FIND_DIRECTION {
    TMCP_FD_BACKWARD,
    TMCP_FD_FORWARD
} TMCP_FIND_DIRECTION;
```

- *TMCP_FD_BACKWARD* - выборка ранней записи.
- *TMCP_FD_FORWARD* - выборка поздней записи.

Объявлено в TMCP.h

4.2.3. TMCP_INCIDENT

Отслеживаемые события и происшествия

```
  typedef enum tagTMCP_INCIDENT {
    TMCP_INC_WRONG_DIRECTION,
    TMCP_INC_SPEED_VIOLATION,
    TMCP_INC_STOPPED_VEHICLE,
    TMCP_INC_JAM_START,
    TMCP_INC_JAM_FINISH
} TMCP_INCIDENT;
```

- *TMCP_INC_WRONG_DIRECTION* - движение по встречной полосе.
- *TMCP_INC_SPEED_VIOLATION* - нарушение скоростного режима.
- *TMCP_INC_STOPPED_VEHICLE* - останов трансп. средства.
- *TMCP_INC_JAM_START* - начало затора.

- *TMCP_INC_JAM_FINISH* - конец затора.

Объявлено в TMCP.h

4.2.4. TMCP_MOV_DIRECTION

Уведомления

```
  typedef enum tagTMCP_MOV_DIRECTION {
    TMCP_MD_FROM_CAMERA,
    TMCP_MD_TO_CAMERA
} TMCP_MOV_DIRECTION;
```

- *TMCP_MD_FROM_CAMERA* - движение от камеры.
- *TMCP_MD_TO_CAMERA* - движение к камере.

Объявлено в TMCP.h

4.2.5. TMCP_NOTIFIFICATION

Уведомления

```
  typedef enum tagTMCP_NOTIFICATION {
    TMCP_NOTIF_VEHICLE,
    TMCP_NOTIF_INCIDENT,
    TMCP_NOTIF_STATISTICS
} TMCP_NOTIFICATION;
```

- *TMCP_NOTIF_VEHICLE* - уведомление об обнаружении ТС.
- *TMCP_NOTIF_INCIDENT* - уведомление о происшествии.
- *TMCP_NOTIF_STATISTICS* - уведомление о фиксации статистики.

Объявлено в TMCP.h

4.2.6. TMCP_RESULT

Результат вызова функций API

```
  typedef enum tagTMCP_ERROR {
    TMCP_RES_OK,
    TMCP_RES_SYSTEM_ERROR,
    TMCP_RES_INVALID_COMMAND,
    TMCP_RES_INVALID_VALUE,
```

```
    TMCP_RES_NO_DATA,
    TMCP_RES_AUTHENTICATION_ERROR
} TMCP_RESULT;
```

- *TMCP_RES_OK* - ет ошибок.
- *TMCP_RES_SYSTEM_ERROR* - ошибка в работе системы не позволяет ответить на запрос или выполнить команду.
- *TMCP_RES_INVALID_COMMAND* - недопустимая команда в текущем режиме работы.
- *TMCP_RES_INVALID_VALUE* - невозможно установить заданное значение.
- *TMCP_RES_NO_DATA* - отсутствуют запрошенные данные.
- *TMCP_RES_AUTHENTICATION_ERROR* - ошибка аутентификации.

Объявлено в TMCP.h

4.2.7. TMCP_REQUEST

Запросы

```
  typedef enum tagTMCP_REQUEST {
    TMCP_REQ_VERSION = 1,
    TMCP_REQ_CLIENT_SERVER_SETTINGS,
    TMCP_REQ_IS_STARTED,
    TMCP_REQ_AFFIXMENT,
    TMCP_REQ_LANES,
    TMCP_REQ_STAT_NOTIF,
    TMCP_REQ_ACC_PERIOD,
    TMCP_REQ_IP_CAM_URI,
    TMCP_REQ_INC_RECORD_LT,
    TMCP_REQ_INC_RECORD_ID,
    TMCP_REQ_STATATISTICS_LT,
    TMCP_REQ_STATISTICS_ID,
    TMCP_REQ_VEHICLE_LT,
    TMCP_REQ_VEHICLE_ID,
} TMCP_REQUEST;
```

- *TMCP_REQ_VERSION* - запрос версии ПО.
- *TMCP_REQ_CLIENT_SERVER_SETTINGS* - запрос клиент-серверных настроек.
- *TMCP_REQ_IS_STARTED* - запрос режима работы TM.

- *TMCP_REQ_AFFIXMENT* - запрос привязки камеры.
- *TMCP_REQ_LANES* - запрос параметров полос движения.
- *TMCP_REQ_STAT_NOTIF* - запрос флага разрешения уведомлений о записи статистики.
- *TMCP_REQ_ACC_PERIOD* - запрос периода накопления статистики.
- *TMCP_REQ_IP_CAM_URI* - запрос URI IP камеры.
- *TMCP_REQ_INC_RECORD_LT* - запрос записи о происшествии по времени.
- *TMCP_REQ_INC_RECORD_ID* - запрос записи о происшествии по идентификатору.
- *TMCP_REQ_STATATISTICS_LT* - запрос записи статистики по времени.
- *TMCP_REQ_STATISTICS_ID* - запрос записи о текущей статистики по идентификатору.
- *TMCP_REQ_VEHICLE_LT* - запрос записи помашинной статистики по времени.
- *TMCP_REQ_VEHICLE_ID* - запрос записи помашинной статистики по идентификатору.

Объявлено в TMCP.h

4.2.8. TMCP_VEH_CLASS

Классы транспортных средств

```
  typedef enum tagTMCP_VEH_CLASS {
    TMCP_VC_MOTORCYCLE,
    TMCP_VC_CAR,
    TMCP_VC_SHORT_TRUCK,
    TMCP_VC_MIDDLE_TRUCK,
    TMCP_VC_LONG_TRUCK,
    TMCP_VC_BUS
} TMCP_VEH_CLASS;
```

- *TMCP_VC_MOTORCYCLE* - мотоцикл.
- *TMCP_VC_CAR* - легковой автомобиль.
- *TMCP_VC_SHORT_TRUCK* - грузовой автомобиль (до 11 м).
- *TMCP_VC_MIDDLE_TRUCK* - грузовой автомобиль (от 11 до 14 м).

- *TMCP_VC_LONG_TRUCK* - грузовой автомобиль (свыше 14 м).
- *TMCP_VC_BUS* - автобус.

Объявлено в TMCP.h

4.2.9. TMCPC_CON_REASON

Коды возврата функций TMCPC

```
  typedef enum tagTMCPC_CON_REASON {
    TMCPC_CON_REASON_CONNECTION,
    TMCPC_CON_REASON_DISCONNECTION
} TMCPC_CON_REASON;
```

- *TMCPC_CON_REASON_CONNECTION* - соединение установлено.
- *TMCPC_CON_REASON_DISCONNECTION* - соединение разорвано.

Объявлено в TMCPC.h

4.2.10. TMCPC_RESULT

Коды возврата функций TMCPC

```
  typedef enum tagTMCPC_RESULT {
    TMCPC_RES_OK,
    TMCPC_RES_INVALID_HANDLE,
    TMCPC_RES_MEMORY_ALLOCATION_ERROR,
    TMCPC_RES_NO_INITIALIZATION,
    TMCPC_RES_NO_CONNECTION,
    TMCPC_RES_INVALID_PARAMETER,
    TMCPC_RES_INVALID_COMMAND,
    TMCPC_RES_INVALID_VALUE,
    TMCPC_RES_NO_DATA,
    TMCPC_RES_AUTHENTICATION_ERROR,
    TMCPC_RES_ANOTHER_ANSWER_IS_RECEIVED,
    TMCPC_RES_SYSTEM_ERROR,
    TMCPC_RES_TIMEOUT,
    TMCPC_RES_UNKNOWN,
    TMCPC_RES_SOCK_INITIALIZATION_ERROR,
    TMCPC_RES_SOCK_CREATION_ERROR,
    TMCPC_RES_SOCK_OPTIONS_SETTING_ERROR,
    TMCPC_RES_SOCK_BINDING_ERROR,
    TMCPC_RES_SOCK_LISTENING_ERROR,
    TMCPC_RES_SOCK_WRITING_ERROR,
    TMCPC_RES_SOCK_UNKNOWN_HOSTNAME
```

```
} TMCPC_RESULT;
```

- *TMCPC_RES_OK* - нет ошибок.
- *TMCPC_RES_INVALID_HANDLE* - хэндл не найден.
- *TMCPC_RES_MEMORY_ALLOCATION_ERROR* - ошибка выделения памяти.
- *TMCPC_RES_NO_INITIALIZATION* - не была вызвана функция инициализации.
- *TMCPC_RES_NO_CONNECTION* - нет соединения.
- *TMCPC_RES_INVALID_PARAMETER* - ошибочно задан параметр.
- *TMCPC_RES_INVALID_COMMAND* - команда не может быть выполнена.
- *TMCPC_RES_INVALID_VALUE* - передано не верное значение на прикладном уровне.
- *TMCPC_RES_NO_DATA* - отсутствуют запрашиваемые данные.
- *TMCPC_RES_AUTHENTICATION_ERROR* - ошибка аутентификации.
- *TMCPC_RES_ANOTHER_ANSWER_IS_RECEIVED* - получен ответ на другой запрос или команду.
- *TMCPC_RES_SYSTEM_ERROR* - ошибка в работе системы.
- *TMCPC_RES_TIMEOUT* - выход по таймауту.
- *TMCPC_RES_UNKNOWN* - получен неизвестный идентификатор ошибки.
- *TMCPC_RES_SOCK_INITIALIZATION_ERROR* - ошибка инициализации сокета.
- *TMCPC_RES_SOCK_CREATION_ERROR* - ошибка создания сокета.
- *TMCPC_RES_SOCK_OPTIONS_SETTING_ERROR* - ошибка установки опций сокета.
- *TMCPC_RES_SOCK_BINDING_ERROR* - ошибка привязки сокета.
- *TMCPC_RES_SOCK_LISTENING_ERROR* - ошибка при переходе сокета в режим приёма.
- *TMCPC_RES_SOCK_WRITING_ERROR* - ошибка записи в сокет.
- *TMCPC_RES_SOCK_UNKNOWN_HOSTNAME* - неизвестное имя хоста.

Объявлено в TMCPC.h

4.3. Идентификаторы

Идентификатор	Значение	Описание
TMCP_ADDR_LEN	256	Максимальная длина задаваемого IP адреса или сетевого имени, включая завершающий ноль.
TMCP_INCIDENTS	5	Количество типов отслеживаемых происшествий
TMCP_IP_CAM_URI_LEN	1024	Максимальная длина URI IP камеры, включая завершающий ноль.
TMCP_MAX_LANES	6	Максимальное количество полос на дороге
TMCP_MIN_ACC_PERIOD_VALUE	5	Минимальное значение периода накопления текущей статистики
TMCP_NOTIFICATION_ID	255	Идентификатор уведомления
TMCP_PASSWORD_LEN	16	Максимальный размер пароля для подключений к TM (включая завершающий нуль)
TMCP_VEH_CLASSES	6	Количество классов ТС

Объявлены в TMCP.h

4.4. Структуры

4.4.1. S_TMCP_Affixment

Привязка камеры к местности

```
  typedef struct tagS_TMCP_Affixment {
    S_TMCP_Point image[4];
    S_TMCP_FPoint earth[4];
    S_TMCP_Point left_edge[2];
    S_TMCP_Point right_edge[2];
} S_TMCP_Affixment;
```

- *image* - координаты опорных точек на изображении (экранные координаты) (пикс).
- *earth* - координаты опорных точек на земле (м).
- *left_edge* - координаты точек отрезка, лежащего на левой границе анализируемого участка дороги (экранные координаты) (пикс).
- *right_edge* - координаты точек отрезка, лежащего на правой границе анализируемого участка дороги (экранные координаты) (пикс).

Примечание:

Основные параметры разметки показаны на рисунке 4.1 (показана разметка для двух полос движения).

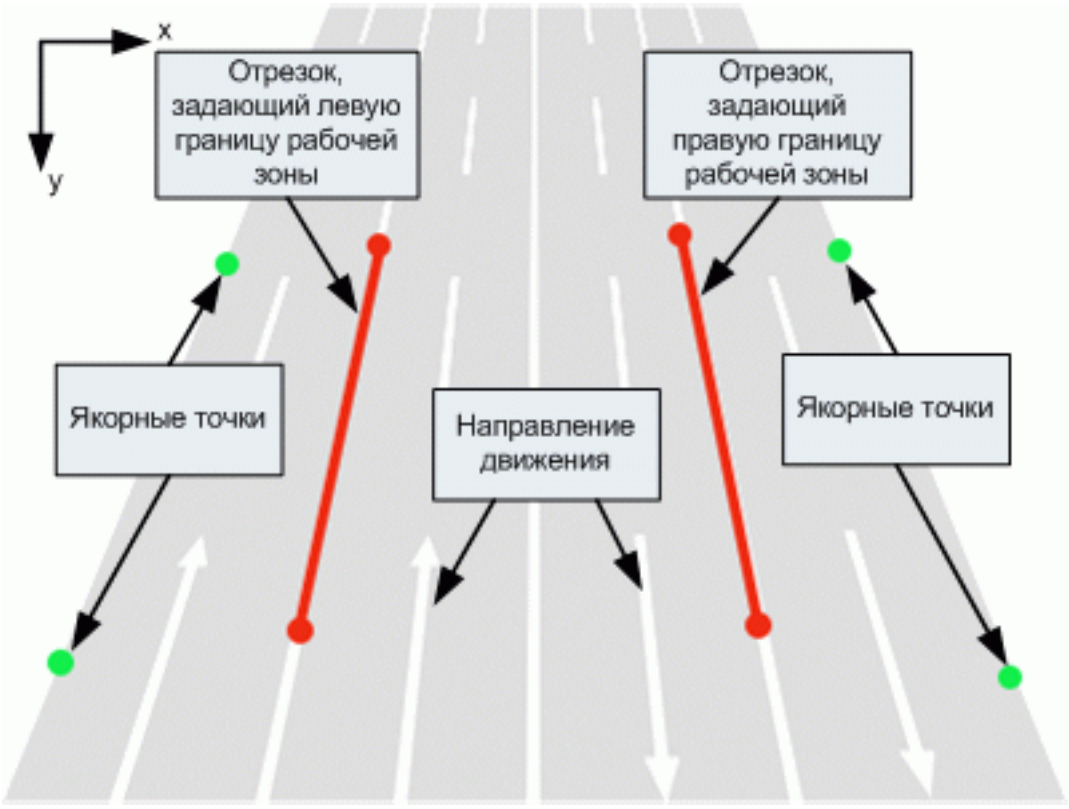

Рисунок 4.1 - Параметры разметки

Опрорные точки задаются в земных (положение точек на плоскости дороги в произвольной Декартовой системе координат) и экранных (положение пикселя) координатах. Точки, задающие отрезки для определения левой и правой границ рабочей зоны, определяются в экранных координатах.

Объявлено в TMCP.h

4.4.2. S_TMCP_ClientServerSettings

Параметры клиент-сервер

```
  typedef struct tagS_TMCP_ClientServerSettings {
    char tm_to_control_addr[TMCP_ADDR_LEN];
    uint16_t control_to_tm_port;
    uint16_t tm_to_control_port;
} S_TMCP_ClientServerSettings;
```

- *tm_to_control_addr* - IP адрес или сетевое имя управляющей станции (УС).
- *control_to_tm_port* - номер TCP порта для присоединений к ТМ.

- *tm_to_control_port* - номер TCP порта для присоединений к УС.

Примечание:

ТМ способен одновременно работать как TCP/IP сервер и как TCP/IP клиент. ТМ ожидает подключение управляющей станции по порту, заданному в поле *tm_to_control_port* (работает как сервер). Если задан *tm_to_control_addr* (не нулевая строка), то ТМ будет одновременно пытаться установить соединение с УС. В поле *tm_to_control_addr* можно задать пустую строку. В этом случае ТМ будет работать только как сервер и не будет пытаться сам установить соединение.

IP адрес формируется следующим образом: номера узлов записываются в 4-х байтовое поле в прямом порядке. Например, для адреса 10.7.1.179 (hex: A.7.1.B3) 4-байтовое значение будет: A0701B3h.

Объявлено в TMCP.h

4.4.3. S_TMCP_FPoint

Координаты точки

```
  typedef struct tagS_TMCP_Point {
    float x;
    float y;
} S_TMCP_FPoint;
```

- *x* - X координата
- *y* - Y координата

Объявлено в TMCP.h

4.4.4. S_TMCP_Incident

Запись о происшествии

```
  typedef struct tagS_TMCP_Incident {
    uint32_t id;
    S_TMCP_TimeStamp time_stamp;
    uint8_t incident;
    uint8_t vehicle;
    uint8_t speed;
} S_TMCP_Incident;
```

- *id* - уникальный идентификатор записи.
- *time_stamp* - время фиксации записи.

- *incident* - тип происшествия. Допустимые значения определены в перечислении TMCP_INCIDENT.
- *vehicle* - для выезда на встречную полосу и превышения скорости содержит идентификатор класса ТС. Для других происшествий не используется.
- *speed* - для (выезда на встречную полосу) и (превышения скорости) содержит значение скорости ТС. Для других происшествий не используется.

Объявлено в TMCP.h

4.4.5. S_TMCP_IncNotif

Уведомление о происшествии

```
  typedef struct tagS_TMCP_IncNotif{
    S_TMCP_Incident incident;
    uint8_t lane;
} S_TMCP_IncNotif;
```

- *incident* - запись о происшествии.
- *lane* - номер полосы движения. Нумерация полос движения начинается с нуля, слева-направо (на изображении).

Объявлено в TMCP.h

4.4.6. S_TMCP_LaneSettings

Параметры полосы движения

```
  typedef struct tagS_TMCP_LaneSettings{
    uint8_t direction;
    uint8_t width;
    uint8_t min_speed;
    uint8_t max_speed;
    uint8_t save_incidents[TMCP_INCIDENTS];
    uint8_t save_statistics;
    uint8_t save_vehicles[TMCP_VEH_CLASSES];
    uint8_t notify_vehicles;
    uint8_t notify_incidents;
    uint8_t jam_speed;
    uint8_t jam_occupancy;
    uint8_t jam_start_time;
    uint8_t jam_finish_time;
} S_TMCP_LaneSettings;
```

- *direction* - Правильное направление движения по полосе. 0 - от камеры, 1 - к камере

- *width* - Ширина полосы в процентах, относительно ширины рабочей зоны (%).
- *min_speed* - Минимально разрешенная скорость (км/ч).
- *max_speed* - Максимально разрешенная скорость (км/ч).
- *save_incidents[TMCP_INC_WRONG_DIRECTION]* - Флаг разрешения фиксации выездов на встречную полосу. 1 - разрешено, 0 - запрещено.
- *save_incidents[TMCP_INC_SPEED_VIOLATION]* - Флаг разрешения фиксации нарушений скоростного режима. 1 - разрешено, 0 - запрещено.
- *save_incidents[TMCP_INC_STOPPED_VEHICLE]* - Флаг разрешения фиксации остановов ТС. 1 - разрешено, 0 - запрещено.
- *save_incidents[TMCP_INC_START_JAM]* - Флаг разрешения фиксации начала затора. 1 - разрешено, 0 - запрещено.
- *save_incidents[TMCP_INC_FINISH_JAM]* - Флаг разрешения фиксации конца затора. 1 - разрешено, 0 - запрещено.
- *save_statistics* - Флаг разрешения фиксации статистики. 1 - разрешено, 0 - запрещено.
- *save_vehicles[TMCP_VC_MOTORCYCLE]* - Флаг разрешения фиксации мотоциклов. 1 - разрешено, 0 - запрещено.
- *save_vehicles[TMCP_VC_CAR]* - Флаг разрешения фиксации легковых. 1 - разрешено, 0 - запрещено.
- *save_vehicles[TMCP_VC_SHORT_TRUCK]* - Флаг разрешения фиксации грузовых до 11 м. 1 - разрешено, 0 - запрещено.
- *save_vehicles[TMCP_VC_MIDDLE_TRUCK]* - Флаг разрешения фиксации грузовых от 11 до 14 м. 1 - разрешено, 0 - запрещено.
- *save_vehicles[TMCP_VC_LONG_TRUCK]* - Флаг разрешения фиксации грузовых свыше 14 м. 1 - разрешено, 0 - запрещено.
- *save_vehicles[TMCP_VC_BUS]* - Флаг разрешения фиксации автобусов. 1 - разрешено, 0 - запрещено.
- *notify_vehicles* - Флаг разрешения уведомлений об онаруженных ТС. 1 - разрешено, 0 - запрещено.
- *notify_incidents* - Флаг разрешения уведомлений об онаруженных происшествиях. 1 - разрешено, 0 - запрещено.
- *jam_speed* - Пороговая скорость для определения затора (км/ч)
- *jam_occupancy* - Пороговая занятость (%) для определения затора
- *jam_start_time* - Период для обнаружения начала затора (сек.)

- *jam_finish_time* - Период для обнаружения конца затора (сек.)

Примечание:

Для обнаружения затора на полосе движения используются следующие признаки:

- В течение периода для обнаружения затора средняя скорость движения не превышает пороговую скорость.
- В течение периода для обнаружения затора интенсивность дорожного движения превышает пороговую интенсивность.

Затор фиксируется только тогда, когда присутствуют оба признака.

Для обнаружения окончания затора на полосе движения можно использовать следующие признаки:

- После обнаружения затора в течение периода для обнаружения конца затора интенсивность дорожного движения не превышает пороговую интенсивность.
- После обнаружения затора в течение периода для обнаружения конца затора средняя скорость движения превышает пороговую скорость.

Окончание затора фиксируется тогда, когда присутствует хотя бы один из признаков.

Объявлено в TMCP.h

4.4.7. S_TMCP_Lanes

Параметры полос движения

```
  typedef struct tagS_TMCP_Lanes{
    uint8_t lanes;
    S_TMCP_LaneSettings lane_settings[TMCP_MAX_LANES];
} S_TMCP_Lanes;
```

- *lanes* - количество полос движения в рабочей зоне.
- *lane_settings* - Параметры каждой из полос движения. Используются только первые *lanes* элементов массива *lane_settings*, параметры остальных элементов игнорируются.

Объявлено в TMCP.h

4.4.8. S_TMCP_Point

Координаты точки

```
  typedef struct tagS_TMCP_Point{
    int16_t x;
```

```
    int16_t y;
} S_TMCP_Point;
```

- *x* - X координата
- *y* - Y координата

Объявлено в TMCP.h

4.4.9. S_TMCP_Statistics

Запись статистики по полосе

```
  typedef struct tagS_TMCP_Statistics {
    uint32_t id;
    S_TMCP_TimeStamp time_stamp;
    uint16_t acc_time;
    uint16_t total_vehicles;
    uint16_t vehicles[TMCP_VEH_CLASSES];
    uint8_t avg_speed;
    uint8_t speed[TMCP_VEH_CLASSES];
    uint8_t avg_msd;
    uint8_t avg_occupancy;
    uint8_t avg_distance;
    uint8_t avg_headway;
    uint16_t incidents[TMCP_INCIDENTS - 1];
}S_TMCP_Statistics;
```

- *id* - уникальный идентификатор записи.
- *time_stamp* - локальное время фиксации записи.
- *acc_time* - время накопления статистики (сек.).
- *total_vehicles* - количество всех обнаруженных транспортных средств.
- *vehicles* - количество обнаруженных ТС по классам. Классы описываются в перечислении TMCP_VEH_CLASS.
- *avg_speed* - средняя скорость всех обнаруженных транспортных средств (км/ч).
- *speed* - средняя скорость ТС по классам (км/ч). Классы описываются в перечислении TMCP_VEH_CLASS.
- *avg_msd* - среднеквадратичное отклонение скорости отдельных ТС от средней скорости всех ТС (км/ч).
- *avg_occupancy* - средняя загруженность (%).

- *avg_distance* - средняя дистанция между транспортными средствами (м).
- *avg_headway* - средний интервал между ТС (сек.).
- *incidents* - число происшествий по типам происшествий. Типы происшествий описываются в перечислении TMCP_INCIDENT.

Примечание:

Средняя загруженность измеряется в области, расположенной в конце рабочей зоны полосы движения (по направлению движения) и имеет протяжённость 3-5 м (см. рис. 4.2). Средняя загруженность определяется как процентное отношение времени, в течение которого ТС находились в измеряемой области, к времени накопления.

Рисунок 4.2 - Области измерения загруженности

Средняя дистанция между ТС - это среднее расстояние между передними бамперами следующих друг за другом ТС.

Средний интервал между ТС - это среднее время, которое необходимо ТС для покрытия дистанции до предыдущего ТС.

Количество заторов показывает сколько раз за период накопления образовывались заторы на полосе движения. ТМ определяет затор, анализируя скорость и загруженность. Если скорость движения ТС ниже некоторого заданного порога скорости, и при этом загруженность выше заданного порога загруженности, то ТМ детектирует начало затора.

Объявлено в TMCP.h

4.4.10. S_TMCP_TimeStamp

Временная метка

```
  typedef struct tagS_TMCP_TimeStamp {
    uint8_t year;
    uint8_t month;
    uint8_t day;
    uint8_t hour;
    uint8_t min;
    uint8_t sec;
    uint16_t msec;
} S_TMCP_TimeStamp;
```

- *year* - Год, начиная с 2000 [0 .. 15].
- *month* - Месяц [1 .. 12].
- *day* - День [1 .. 31].
- *hour* - Час [0 .. 23].
- *min* - Минуты [0 .. 59].
- *sec* - Секунды [0 .. 59].
- *ushMSec* - Миллисекунды [0 .. 999].

Объявлено в TMCP.h

4.4.11. S_TMCP_Vehicle

Параметры обнаруженного ТС

```
  typedef struct tagS_TMCP_Vehicle{
    uint32_t id;
    S_TMCP_TimeStamp time_stamp;
    uint8_t vehicle;
    uint8_t speed;
    uint8_t length;
    uint8_t distance;
    uint8_t headway;
    uint16_t pos_x;
    uint16_t pos_y;
} S_TMCP_Vehicle;
```

- *id* - идентификатор записи.
- *time_stamp* - абсолютное время обнаружения ТС.

- *vehicle* - класс транспортного средства. Допустимые значения определены в перечислении TMCP_VEH_CLASS.
- *speed* - скорость транспортного средства (км/ч).
- *length* - длина ТС (м).
- *distance* - расстояние между передними бамперами обнаруженного ТС и предыдущего ТС (м).
- *headway* - интервал между обнаруженным ТС и предыдущим ТС (с).
- *pos_x* - X координата на изображении центра прямоугольника, описанного вокруг ТС (пикс.).
- *ushScreenY* - Y координата на изображении центра прямоугольника, описанного вокруг ТС (пикс.).

Объявлено в TMCP.h

4.4.12. S_TMCP_VehNotif

Уведомление о ТС

```
  typedef struct tagS_TMCP_VehNotif{
    S_TMCP_Vehicle vehicle;
    uint8_t lane;
} S_TMCP_VehNotify;
```

- *vehicle* - запись об обнаруженном ТС.
- *lane* - номер полосы движения. Нумерация полос движения начинается с нуля, слева-направо.

Объявлено в TMCP.h

4.4.13. S_TMCPC_ConParams

Параметры установки соединения с устройством ТМ

```
  typedef struct tagS_TMCPC_ConParams{
    uint8_t listen;
    char addr[256];
    uint16_t port_num;
    uint32_t request_timeout;
    TMCPC_CONNECTION_HANDLER connection_handler;
    TMCPC_NOTIFICATION_HANDLER notification_handler;
    void *connection_handler_param;
    void *notification_handler_param;
```

```
} S_TMCPC_ConParams;
```

- *listen* - управляющая станция ожидает подключения (работает ка сервер).
- *addr* - IP адрес или сетевое имя устройства TM.
- *port_num* - номер TCP порта.
- *request_timeout* - время, отводимое на выполнение запроса или команды (сек.).
- *connection_handler* - функция - обработчик событий "соединение" и "разъединение".
- *notification_handler* - функция - обработчик уведомлений.
- *connection_handler_param* - параметр, передаваемый в обработчик событий "соединение" и "разъединение".
- *notification_handler_param* - параметр, передаваемый в обработчик уведомлений.

Примечание:

Если УС работает как сервер и ожидает подключения (поле *listen* = 1), то поле *addr* игнорируется, так как адрес клиента-устройства TM заранее неизвестен. Если *listen* = 0, то УС будет инициировать соединение и необходимо задать адрес устройства *addr*.

Объявлено в TMCPC.h

4.4.14. U_TMCP_IPAddress

Представление IP адреса и версии ПО

```
  typedef union tagU_TMCP_IPAddress {
    uint32_t val32;
    uint8_t val8[sizeof(uint32_t)];
} U_TMCP_IPAddress, U_TMCP_Version;
```

- *val32* - IP адрес или версия ПО в 4-байтовом слове.
- *val8* - IP адрес или версия ПО в 4-х байтовом массиве.

Объявлено в TMCP.h

4.5. Функции

4.5.1. Функции для управления устройством TM

4.5.1.1. TMCPC_Authentication

Аутентификация администратора устройства TM

```
TMCPC_RESULT TMCPC_Authentication(
   TMCPC_HANDLE tmcpc,
   const char *password
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

- *password* - [in] ASCII строка с завершающим нулём. Содержит пароль для администрирования устройства TM. Максимальный размер строки с учётом завершающего нуля - TMCP_PASSWORD_SIZE символов.

Возвращаемое значение: идентификатор ошибки.

Примечание:

В каждом сеансе связи, сразу после установки соединения с устройством TM и перед тем, как начать администрирование необходимо пройти аутентификацию, то есть передать пароль. При задании пароля допускается использование только печатных символов без пробелов и табуляции.

Функция возвращает управление либо при возникновении ошибки, либо после окончания аутентификации, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример использования функции см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.2. TMCPC_ChangePassword

Изменение пароля администратора

```
TMCPC_RESULT TMCPC_ChangePassword(
   TMCPC_HANDLE tmcpc,
   const char *new_password
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

- *new_password* - [in] ASCII строка с завершающим нулём. Содержит новый пароль для администрирования устройства TM. Максимальный размер строки с учётом завершающего нуля - TMCP_PASSWORD_SIZE символов.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Новый пароль администратора всупит в силу в следующем сеансе связи. При задании пароля допускается использование только печатных символов без пробелов и табуляции.

Функция возвращает управление либо при возникновении ошибки, либо после окончания установки нового пароля, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.3. TMCPC_ClearData

Удаление накопленных на устройстве данных.

```
TMCPC_RESULT TMCPC_ClearData(
   TMCPC_HANDLE tmcpc
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

В процессе очистки базы данных удаляются:

- Статистика.
- Помашинная статистика.
- Информация о происшествиях.

Функция возвращает управление либо при возникновении ошибки, либо после окончания, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.4. TMCPC_Connect

Присоединение к устройству TM.

```
TMCPC_RESULT TMCPC_Connect(
   TMCPC_HANDLE tmcpc,
   const S_TMCPC_ConParams *con_params
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *con_params* - [in] параметры соединения.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Для разрыва соединения используйте вызов TMCPC_Disconnect.

Функция возвращает управление не дожидаясь установки соединения. Для проверки существования соединения можно периодически выполнять вызов TMCPC_IsConnect.

Пример 1 установки соединения:

```
  #ifdef _WIN32
#include <windows.h>
#define SLEEP(MS) Sleep((MS))
#else
#include <unistd.h>
#define SLEEP(MS) usleep((MS) * 1000)
#endif
#include <stdio.h>
#include <string.h>
#include "tmcpc.h"

static TMCPC_HANDLE Connect(
        const S_TMCPC_ConParams *con_params,
        const char *password) {
    TMCPC_HANDLE tmcpc = 0;
    U_TMCP_IPAddress addr;
    TMSPS_RESULT res;

        // Creation and initialization...
    res = TMCPC_Create(&tmcpc);
    if(TMCPC_RES_OK != res) {
            // Error handling...
        return 0;
    }
    res = TMCPC_Init(tmcpc);
    if(TMCPC_RES_OK != res) {
            // Error handling...
        TMCPC_Destroy(tmcpc);
        return 0;
    }

        // Wait for connection...
```

```
    res =  TMCPC_Connect(tmcpc, con_params);
    if(TMCPC_RES_OK != Res) {
            // Error handling...
        TMCPC_Release(tmcpc);
        TMCPC_Destroy(tmcpc);
        return 0;
    }
    while(!TMCPC_IsConnect(tmcpc, &addr))
        SLEEP(1000);

    printf("Connection is established. "
            "Address = %u.%u.%u.%u\n",
            addr.val8[3], addr.val8[2],
            addr.val8[1], addr.val8[0]);

        // Do authentication...
    res = TMCPC_Authentication(tmcpc, password);
    if(TMCPC_RES_OK != res) {
            // Error handling...
        TMCPC_Disconnect(tmcpc);
        TMCPC_Release(tmcpc);
        TMCPC_Destroy(tmcpc);
        return 0;
    }

    return tmcpc;
}

int main() {
    S_TMCPC_ConParams con_params = {0};
    TMCPC_HANDLE tmcpc = 0;

    strcpy(con_params.addr, "10.7.8.25");
    con_params.port_num = 52224;
    con_params.request_timeout = 10;

    tmcpc = Connect(&con_params, "tm");

    TMCPC_Disconnect(tmcpc);
    TMCPC_Release(tmcpc);
    TMCPC_Destroy(tmcpc);
}
```

Альтернативным способом определения соединения является использование функции обратного вызова. Прототип этой функции описан в TMCPC_CONNECTION_HANDLER.

Пример 2 установки соединения (для OS Windows):

```
  #include <windows.h>
#include <stdio.h>
#include <string.h>
#include "tmcpc.h"

static TMCPC_HANDLE Connect(
        const S_TMCPC_ConParams *con_params,
        const char *password) {
    TMCPC_HANDLE tmcpc = 0;
    U_TMCP_IPAddress addr;
    TMSPS_RESULT res;

        // Creation and initialization...
    res = TMCPC_Create(&tmcpc);
    if(TMCPC_RES_OK != res) {
            // Error handling...
        return 0;
    }
    res = TMCPC_Init(tmcpc);
    if(TMCPC_RES_OK != res) {
            // Error handling...
        TMCPC_Destroy(tmcpc);
        return 0;
    }

        // Wait for connection...
    res =  TMCPC_Connect(tmcpc, con_params);
    if(TMCPC_RES_OK != res) {
            // Error handling...
        TMCPC_Release(tmcpc);
        TMCPC_Destroy(tmcpc);
        return 0;
    }
    DWORD wait_res = WaitForSingleObject(
            (HANDLE)con_params->connection_handler_param,
            INFINITE);
    if(WAIT_OBJECT_0 != wait_res) {
        printf("Wait for connection error\n");
        TMCPC_Release(tmcpc);
        TMCPC_Destroy(tmcpc);
        return 0;
    }

        // Connection is established.
        // Do authentication...
    res = TMCPC_Authentication(tmcpc, password);
    if(TMCPC_RES_OK != res) {
```

```
            // Error handling...
        TMCPC_Disconnect(tmcpc);
        TMCPC_Release(tmcpc);
        TMCPC_Destroy(tmcpc);
        return 0;
    }

    return tmcpc;
}

static void __stdcall ConnectionHandler(
        TMCPC_CON_REASON reason,
        U_TMCP_IPAddress addr,
        void *handler_param) {
    if(TMCPC_CON_REASON_CONNECTION == reason) {
        printf("On connection. Address = %u.%u.%u.%u\n",
                addr.val8[3], addr.val8[2],
                addr.val8[1], addr.val8[0]);
        SetEvent((HANDLE)handler_param);
    }
}

void main() {
    S_TMCPC_ConParams con_params = {0};
    TMCPC_HANDLE tmcpc = 0;

    strcpy(con_params.addr, "10.7.8.25");
    con_params.port_num = 52224;
    con_params.request_timeout = 10;
    con_params.connection_handler = ConnectionHandler;
    con_params.connection_handler_param =
            CreateEvent(NULL, FALSE, FALSE, NULL);

    tmcpc = Connect(&con_params, "tm");

    TMCPC_Disconnect(tmcpc);
    TMCPC_Release(tmcpc);
    TMCPC_Destroy(tmcpc);
    CloseHandle((HANDLE)con_params.connection_handler_param);
}
```

Объявлено в TMCPC.h

4.5.1.5. TMCPC_Create

Создание экземпляра объекта TMCPC.

```
  TMCPC_RESULT TMCPC_Create(
```

```
    TMCPC_HANDLE *tmcpc
    );
```

- *tmcpc* - [out] описатель (хэндл) экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция TMCPC_Create должна быть вызвана первой, до остальных вызовов функций библиотеки. Большинство функций TMCPC (кроме функций для работы с временной меткой и функции получения версии API) требуют передачи в качестве параметра ранее полученный описатель. Если описатель не был получен, то функции, требующие описатель, будут возвращать ошибку TMCPC_RES_INVALID_HANDLE. Для удаления ранее созданного объекта TMCPC необходимо вызвать функцию TMCPC_Destroy.

Пример получения описателя объекта TMCPC:

```
  ...
TMCPC_HANDLE tmcpc;
TMCPC_RESULT res = TMCPC_Create(&tmcpc);
if(TMCPC_RES_OK != res){
    // Error handling.
}
else{
        // Initialization
    res = TMCPC_Init(tmcpc);
    if(TMCPC_RES_OK != res){
        // Error handling.
    }
    else{
            // Any working. The other TMCPC functions call.
        ...
            // Deinitialization.
        TMCPC_Release(tmcpc);
    }
        // End of work. Delete the TMCPC object.
    TMCPC_Destroy(tmcpc);
}
```

Объявлено в TMCPC.h

4.5.1.6. TMCPC_Destroy

Удаление экземпляра объекта TMCPC.

```
  TMCPC_RESULT TMCPC_Destroy(
    TMCPC_HANDLE tmcpc
```

```
    );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

После вызова TMCPC_Destroy описатель объекта становится недействительным и не может дальше использоваться.

Объявлено в TMCPC.h

4.5.1.7. TMCPC_Disconnect

Разрыв соединения с устройством TM.

```
TMCPC_RESULT TMCPC_Disconnect(
  TMCPC_HANDLE tmcpc
  );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функцию можно безопасно вызывать, даже если соединение не было установлено.

Объявлено в TMCPC.h

4.5.1.8. TMCPC_GetAccPeriod

Запрос периода накопления статистики.

```
TMCPC_RESULT TMCPC_GetAccPeriod(
  TMCPC_HANDLE tmcpc,
  uint16_t *acc_period
  );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *acc_period* - [out] период накопления статистики.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.9. TMCPC_GetAffixment

Запрос привязки камеры к местности.

```
TMCPC_RESULT TMCPC_GetAffixment(
   TMCPC_HANDLE tmcpc,
   S_TMCP_Affixment *affixment
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *affixment* - [out] привязка.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в lRequestTimeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.10. TMCPC_GetAPIVersion

Получение версии текущего API.

```
TMCPC_RESULT TMCPC_GetAPIVersion(
   uint8_t *tmcp_version,
   uint8_t *tmcpc_version
   );
```

- *tmcp_version* - [out] номер версии протокола TMCP.
- *tmcpc_version* - [out] номер реализации протокола TMCP.

Возвращаемое значение: идентификатор ошибки.

Объявлено в TMCPC.h

4.5.1.11. TMCPC_GetClientServerSettings

Получение настроек клиент-сервер.

```
TMCPC_RESULT TMCPC_GetClientServerSettings(
   TMCPC_HANDLE tmcpc,
   S_TMCP_ClientServerSettings *settings
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *settings* - [out] настройки клиент-сервер.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.12. TMCPC_GetStatisticsID

Запрос данных текущей статистики по идентификатору.

```
TMCPC_RESULT TMCPC_GetStatisticsID(
   TMCPC_HANDLE tmcpc,
   uint32_t id,
   TMCP_FIND_DIRECTION dir,
   uint32_t lane,
   S_TMCP_Statistics *statistics
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *id* - [in] уникальный идентификатор записи.
- *dir* - [in] направление поиска при выборке записи.
- *lane* - [in] номер опрашиваемой полосы движения. Нумерация полос движения начинается с нуля, слева-направо.
- *statistics* - [out] данные статистики.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Статистика на устройстве TM хранится в циклической очереди, которая размещается в файле. Размер очереди по каждой из полос движения задаётся размером файла базы данных и может быть изменён в файле конфигурации. При полностью заполненной очереди новая (вносимая) запись затирает самую старую запись. Период внесения записей, или время накопления статистики задаётся администратором при настройке камеры (см. TMCPC_SetAccPeriod).

Если значение идентификатора, указанного в передаваемом параметре, точно соответствует идентификатору записи, то будет передана эта запись. В остальных случаях алгоритм выборки определяется значением параметра Dir.

Значение параметра Dir	**Выборка**
TMCP_FD_FORWARD	Выбирается ближайшая запись с большим значением идентификатора (направление поиска - вперёд)
TMCP_FD_BACKWARD	Выбирается ближайшая запись с меньшим значением идентификатора (направление поиска - назад)

В случае отсутствия запрашиваемой записи функция возвращает ошибку TMCPC_RES_NO_DATA.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример запроса 10-ти последних записей первой полосы движения:

```
  ...
uint32_t id;
S_TMCP_Statistics record[10];
TMCPC_RESULT res;
int i;

id = 0xFFFFFFFF;
for(i=9; i>=0; i--){
    res = TMCPC_GetStatisticsID(tmcpc, id,
        TMCP_FD_BACK, 0, &record[i]);
    if(TMCPC_RES_OK != res){
        break;
    }
    else{
        id = Record[i].id - 1;
    }
}
...
```

Пример предварительного подключения к ТМ см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.13. TMCPC_GetStatisticsLT

Запрос данных статистики по метке времени.

```
TMCPC_RESULT TMCPC_GetStatisticsLT(
   TMCPC_HANDLE tmcpc,
   S_TMCP_TimeStamp *time_stamp,
   TMCP_FIND_DIRECTION dir,
   uint8_t lane,
   S_TMCP_Statistics *statistics
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *time_stamp* - [in] локальное время, которому соответствует запрашиваемые данные.
- *dir* - [in] направление поиска при выборке записи.
- *lane* - [in] номер опрашиваемой полосы движения. Нумерация полос движения начинается с нуля, слева-направо.
- *statistics* - [out] данные статистики (версия 3).

Возвращаемое значение: идентификатор ошибки.

Примечание:

Статистика на устройстве ТМ хранится в циклической очереди, которая размещается в файле. Размер очереди по каждой из полос движения задаётся размером файла базы данных и может быть изменён в файле конфигурации. При полностью заполненной очереди новая (вносимая) запись затирает самую старую запись. Период внесения записей, или время накопления статистики задаётся администратором при настройке камеры (см. TMCPC_SetAccPeriod).

Если время, указанное в передаваемой временной метке, точно соответствует времени записи, то будет передана эта запись. В остальных случаях алгоритм выборки определяется значением параметра Dir.

Значение параметра Dir	**Выборка**
TMCP_FD_FORWARD	Выбирается наиболее РАННЯЯ запись, внесённая ПОСЛЕ указанного времени (направление поиска - вперёд)

Значение параметра Dir	Выборка
TMCP_FD_BACKWARD	Выбирается наиболее ПОЗДНЯЯ запись, внесённая ДО указанного времени (направление поиска - назад)

В случае отсутствия запрашиваемой записи функция возвращает ошибку TMCPC_RES_NO_DATA.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример запроса 10-ти последних записей первой полосы движения:

```
  ...
S_TMCP_TimeStamp time_stamp;
S_TMCP_Statistics record[10];
TMCPC_RESULT res;
int i;

    // Get the max time stamp
TMCPC_TS_GetMaxTime(&time_stamp);

for(i=9; i>=0; i--){
    Res = TMCPC_GetStatisticsLT(tmcpc,
        &time_stamp, TMCP_FD_BACK, 0, &record[i]);
    if(TMCPC_RES_OK != res){
        break;
    }
    else{
        TMCPC_TS_DecTime(&record[i].time_stamp,
            &time_stamp);
    }
}
...
```

Пример предварительного подключения к TM см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.14. TMCPC_GetIncidentID

Запрос записи о происшествии по идентификатору.

```
  TMCPC_RESULT TMCPC_GetIncidentID(
    TMCPC_HANDLE tmcpc,
    uint32_t id,
    TMCP_FIND_DIRECTION dir,
```

```
    uint8_t lane,
    S_TMCP_Incident *incident
    );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *id* - [in] уникальный идентификатор записи.
- *dir* - [in] направление поиска при выборке записи.
- *lane* - [in] номер опрашиваемой полосы движения. Нумерация полос движения начинается с нуля, слева-направо.
- *incident* - [out] запись о происшествии.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Если значение идентификатора, указанного в передаваемом параметре, точно соответствует идентификатору записи, то будет передана эта запись. В остальных случаях алгоритм выборки определяется значением параметра Dir.

Значение параметра Dir	**Выборка**
TMCP_FD_FORWARD	Выбирается ближайшая запись с большим значением идентификатора (направление поиска - вперёд)
TMCP_FD_BACKWARD	Выбирается ближайшая запись с меньшим значением идентификатора (направление поиска - назад)

В случае отсутствия запрашиваемой записи функция возвращает ошибку TMCPC_RES_NO_DATA.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример запроса 10-ти последних записей для второй полосы движения:

```
  ...
uint32_t id;
S_TMCP_Incident record[10];
TMCPC_RESULT res;
int i;
```

```
id = 0xFFFFFFFF;
for(i=9; i>=0; i--){
    Res = TMCPC_GetIncidentID(tmcpc, id,
        TMCP_FD_BACK, 1, &record[i]);
    if(TMCPC_RES_OK != res){
        break;
    }
    else{
        id = record[i].id - 1;
    }
}
...
```

Пример предварительного подключения к TM см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.15. TMCPC_GetIncidentLT

Запрос записи о происшествии по метке времени.

```
TMCPC_RESULT TMCPC_GetIncidentLT(
   TMCPC_HANDLE tmcpc,
   S_TMCP_TimeStamp *time_stamp,
   TMCP_FIND_DIRECTION dir,
   uint8_t lane,
   S_TMCP_Incident *incident
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *time_stamp* - [in] локальное время запрашиваемой записи о происшествии.
- *dir* - [in] направление поиска при выборке записи.
- *lane* - [in] номер опрашиваемой полосы движения. Нумерация полос движения начинается с нуля, слева-направо.
- *incident* - [out] запись о происшествии.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Если время, указанное в передаваемой временной метке, точно соответствует времени записи, то будет передана эта запись. В остальных случаях алгоритм выборки определяется значением параметра Dir.

Значение параметра Dir	Выборка
TMCP_FD_FORWARD	Выбирается наиболее РАННЯЯ запись, внесённая ПОСЛЕ указанного времени (направление поиска - вперёд)
TMCP_FD_BACKWARD	Выбирается наиболее ПОЗДНЯЯ запись, внесённая ДО указанного времени (направление поиска - назад)

В случае отсутствия запрашиваемой записи функция возвращает ошибку TMCPC_RES_NO_DATA.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример запроса 10-ти последних записей для второй полосы движения:

```
  ...
S_TMCP_TimeStamp time_stamp;
S_TMCP_Incident record[10];
TMCPC_RESULT res;
int i;

    // Get the max time stamp
TMCPC_TS_GetMaxTime(&time_stamp);

for(i=9; i>=0; i--){
    Res = TMCPC_GetIncidentLT(tmcpc,
        &time_stamp, TMCP_FD_BACK, 0, &record[i]);
    if(TMCPC_RES_OK != res){
        break;
    }
    else{
        TMCPC_TS_DecTime(&record[i].time_stamp,
            &time_stamp);
    }
}
...
```

Пример предварительного подключения к TM см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.16. TMCPC_GetIPCamURI

Получение URI IP камеры.

```
  TMCPC_RESULT TMCPC_GetIPCamURI(
```

```
    TMCPC_HANDLE tmcpc,
    char *ip_cam_uri
    );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *ip_cam_uri* - [out] строка с URI IP камеры. Размер строки - не менее TMCP_IP_CAM_URI_LEN символов.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.17. TMCPC_GetLanes

Получение параметров полос движения.

```
TMCPC_RESULT TMCPC_GetLanes(
   TMCPC_HANDLE tmcpc,
   S_TMCP_Lanes *lanes
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *lanes* - [out] параметры полос движения.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.18. TMCPC_GetVehicleID

Запрос записи об обнаруженном транспортном средстве по идентификатору.

```
TMCPC_RESULT TMCPC_GetVehicleID(
```

```
    TMCPC_HANDLE tmcpc,
    uint32_t id,
    TMCP_FIND_DIRECTION dir,
    uint8_t lane,
    S_TMCP_Vehicle *vehicle
    );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *id* - [in] уникальный идентификатор записи.
- *dir* - [in] направление поиска при выборке записи.
- *lane* - [in] номер опрашиваемой полосы движения. Нумерация полос движения начинается с нуля, слева-направо.
- *vehicle* - [out] запись о транспортном средстве.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Если значение идентификатора, указанного в передаваемом параметре, точно соответствует идентификатору записи, то будет передана эта запись. В остальных случаях алгоритм выборки определяется значением параметра Dir.

Значение параметра Dir	**Выборка**
TMCP_FD_FORWARD	Выбирается ближайшая запись с большим значением идентификатора (направление поиска - вперёд)
TMCP_FD_BACKWARD	Выбирается ближайшая запись с меньшим значением идентификатора (направление поиска - назад)

В случае отсутствия запрашиваемой записи функция возвращает ошибку TMCPC_RES_NO_DATA.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример запроса 10-ти последних записей для второй полосы движения:

```
  ...
uint32_t id;
S_TMCP_Vehicle record[10];
TMCPC_RESULT res;
int i;
```

```
id = 0xFFFFFFFF;
for(i=9; i>=0; i--){
    Res = TMCPC_GetVehicleID(tmcpc, id,
        TMCP_FD_BACK, 1, &record[i]);
    if(TMCPC_RES_OK != res){
        break;
    }
    else{
        id = record[i].id - 1;
    }
}
...
```

Пример предварительного подключения к TM см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.19. TMCPC_GetVehicleLT

Запрос записи об обнаруженном транспортном средстве по метке времени.

```
TMCPC_RESULT TMCPC_GetVehicleLT(
   TMCPC_HANDLE tmcpc,
   S_TMCP_TimeStamp *time_stamp,
   TMCP_FIND_DIRECTION dir,
   uint8_t lane,
   S_TMCP_Vehicle *vehicle
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *time_stamp* - [in] локальное время запрашиваемой записи о происшествии.
- *dir* - [in] направление поиска при выборке записи.
- *lane* - [in] номер опрашиваемой полосы движения. Нумерация полос движения начинается с нуля, слева-направо.
- *vehicle* - [out] запись о транспортном средстве.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Если время, указанное в передаваемой временной метке, точно соответствует времени записи, то будет передана эта запись. В остальных случаях алгоритм выборки определяется значением параметра Dir.

Значение параметра Dir	Выборка
TMCP_FD_FORWARD	Выбирается наиболее РАННЯЯ запись, внесённая ПОСЛЕ указанного времени (направление поиска - вперёд)
TMCP_FD_BACKWARD	Выбирается наиболее ПОЗДНЯЯ запись, внесённая ДО указанного времени (направление поиска - назад)

В случае отсутствия запрашиваемой записи функция возвращает ошибку TMCPC_RES_NO_DATA.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример запроса 10-ти последних записей для второй полосы движения:

```
  ...
S_TMCP_TimeStamp time_stamp;
S_TMCP_Vehicle record[10];
TMCPC_RESULT res;
int i;

    // Get the max time stamp
TMCPC_TS_GetMaxTime(&time_stamp);

for(i=9; i>=0; i--){
    Res = TMCPC_GetVehicleLT(tmcpc,
        &time_stamp, TMCP_FD_BACK, 0, &record[i]);
    if(TMCPC_RES_OK != res){
        break;
    }
    else{
        TMCPC_TS_DecTime(&record[i].time_stamp,
            &time_stamp);
    }
}
...
```

Пример предварительного подключения к TM см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.20. TMCPC_GetVersion

Запрос версии программы устройства.

```
  TMCPC_RESULT TMCPC_GetVersion(
```

```
    TMCPC_HANDLE tmcpc,
    U_TMCP_Version *version
    );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *version* - [out] версия программы TM.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.21. TMCPC_Init

Функция инициализации.

```
TMCPC_RESULT TMCPC_Init(
   TMCPC_HANDLE tmcpc
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция инициализации должна быть вызвана после получения описателя (см. TMCPC_Create) и до остальных вызовов TMCPC API.

Пример использования функции см. в описании TMCPC_Create.

Объявлено в TMCPC.h

4.5.1.22. TMCPC_IsConnect

Проверка наличия соединения.

```
unsigned char TMCPC_IsConnect(
   TMCPC_HANDLE tmcpc,
   U_TMCP_IPAddress *addr
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *addr* - [out] IP адрес устройства TM.

Возвращаемое значение: 1 - соединение установлено, 0 - соединение отсутствует.

Примечание:

Рекомендуется вызывать эту функцию периодически, после вызова TMCPC_Connect для определеления момента окончательной установки соединения с устройством TM.

Параметр pulIPAddress устанавливается в функции только при наличии соединения, то есть когда функция TMCP_IsConnect возвращает 1. Если получение IP адреса не требуется, то вместо этого параметра можно передать нулевое значение.

IP адрес формируется следующим образом: номера узлов записываются в 4-х байтовое поле в прямом порядке. Например, для адреса 10.7.1.179 (hex: A.7.1.B3) 4-байтовое значение будет: A0701B3h.

Пример использования функции см. в описании TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.23. TMCPC_IsStarted

Получения флага режима работы устройства.

```
TMCPC_RESULT TMCPC_IsStarted(
   TMCPC_HANDLE tmcpc,
   uint8_t *is_started
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *is_started* - [out] флаг режима работы. 1 - TM работает и анализирует трафик дорожного движения, 0 - TM находится в настроечном режиме и анализ трафика не производится.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо по окончании своей работы, либо при возникновении ошибки, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.24. TMCPC_IsStatNotif

Получение флага уведомления о записи статистики.

```
TMCPC_RESULT TMCPC_IsStatNotif(
   TMCPC_HANDLE tmcpc,
   uint8_t *notif
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *notif* - [out] флаг уведомления о записи статистики. 0 - уведомления не высылаются, 1 - уведомления высылаются.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.25. TMCPC_Release

Деинициализация.

```
TMCPC_RESULT TMCPC_Release(
   TMCPC_HANDLE tmcpc
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функцию деинициализации можно безопасно вызывать даже если не была вызвана функция инициализации (см. TMCPC_Init), или если инициализация закончилась ошибкой.

Объявлено в TMCPC.h

4.5.1.26. TMCPC_SetAccPeriod

Установка периода накопления статистики.

```
TMCPC_RESULT TMCPC_SetAccPeriod(
   TMCPC_HANDLE tmcpc,
   uint16_t acc_period
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *acc_period* - [in] период накопления статистики.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Допустимый диапазон значений для периода накопления - от 5 до 65535 секунд.

Установка периода накопления возможна только в настроечном режиме работы устройства ТМ.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.27. TMCPC_SetAffixment

Установка привязки камеры к местности.

```
TMCPC_RESULT TMCPC_SetAffixment(
   TMCPC_HANDLE tmcpc,
   const S_TMCP_Affixment *affixment
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *affixment* - [in] привязка.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Установка разметки возможна только в настроечном режиме работы устройства ТМ.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в lRequestTimeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.28. TMCPC_SetClientServerSettings

Установка настроек клиент-сервер.

```
TMCPC_RESULT TMCPC_SetClientServerSettings(
   TMCPC_HANDLE tmcpc,
   const S_TMCP_ClientServerSettings *settings
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *settings* - [in] настройки клиент-сервер.

Возвращаемое значение: идентификатор ошибки.

Примечание:

При задании номеров портов (поля *settings->control_to_tm_port* и *settings->tm_to_control_port* структуры S_TMCP_ClientServerSettings) следует иметь в виду, что некоторые номера могут быть заблокированы брандмауэрами, антивирусами или использоваться для других соединений.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.29. TMCPC_SetIPCamURI

Установка URI IP камеры.

```
TMCPC_RESULT TMCPC_SetIPCamURI(
   TMCPC_HANDLE tmcpc,
   const char *ip_cam_uri
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *ip_cam_uri* - [out] строка с URI IP камеры. Размер строки - не более TMCP_IP_CAM_URI_LEN символов вместе с завершающим нулём.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Установка URI возможна только в настроечном режиме работы устройства TM.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.30. TMCPC_SetLanes

Установка параметров полос движения.

```
TMCPC_RESULT TMCPC_SetLanes(
   TMCPC_HANDLE tmcpc,
   const S_TMCP_Lanes *lanes
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *lanes* - [in] параметры полос движения.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Установка параметров полос движения возможна только в настроечном режиме работы устройства TM.

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.31. TMCPC_SetStatNotif

Разрешение/запрет уведомлений о сохранении статистики.

```
TMCPC_RESULT TMCPC_SetStatNotif(
   TMCPC_HANDLE tmcpc,
   uint8_t notif
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.
- *notif* - [in] флаг разрешения уведомлений. 1 - уведомления будут высылаться, 1 - уведомления не будут высылаться.

Возвращаемое значение: идентификатор ошибки.

Примечание:

Функция возвращает управление либо при возникновении ошибки, либо после получения запрашиваемых данных, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример разрешения и обработки уведомлений о записи статистики см. в разделе TMCPC_NOTIFICATION_HANDLER

Функцию Connect см. в разделе TMCPC_Connect.

Объявлено в TMCPC.h

4.5.1.32. TMCPC_Start

Переключение TM в рабочий режим.

```
TMCPC_RESULT TMCPC_Start(
   TMCPC_HANDLE tmcpc
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

В рабочем режиме TM анализирует трафик дорожного движения, формирует статистику и уведомления. Для старта необходимо задать привязку камеры к местности (см. TMCPC_SetAffixment) и параметры полос движения (см. TMCPC_SetLanes). В рабочем режиме невозможны изменения следующих настроек:

- Изменение привязки камеры к местности (TMCPC_SetAffixment).
- Изменение параметров полос движения (TMCPC_SetLanes).
- Изменение периода накопления статистики (TMCPC_SetAccPeriod).
- Изменение URI IP камеры (TMCPC_SetIPCamURI).

Функция возвращает управление либо при возникновении ошибки, либо после окончания, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.1.33. TMCPC_Stop

Переключение TM в настроечный режим.

```
TMCPC_RESULT TMCPC_Stop(
   TMCPC_HANDLE tmcpc
   );
```

- *tmcpc* - [in] описатель ранее созданного вызовом TMCPC_Create экземпляра объекта TMCPC.

Возвращаемое значение: идентификатор ошибки.

Примечание:

В настроечном режиме трафик дорожного движения не анализируется и уведомления не передаются. Такой режим служит для настройки ТМ и его последующего включения (см. TMCPC_Start).

Функция возвращает управление либо при возникновении ошибки, либо после окончания, либо после истечения времени, заданным в request_timeout структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.5.2. Функции для работы с временной меткой

4.5.2.1. TMCPC_TS_AddTime

Увеличение (уменьшение) значения временной метки.

```
unsigned char TMCPC_TS_AddTime(
   const S_TMCP_TimeStamp *ts,
   int64_t ms,
   S_TMCP_TimeStamp *ts_res
   );
```

- *ts* - [in] увеличиваемое время.
- *ms* - [in] добавляемое время в миллисекундах.
- *ts_res* - [out] результат.

Возвращаемое значение: 1 - нет ошибок, 0 - ошибка.

Примечание:

Функция увеличивает/уменьшает значение временной метки на количество миллисекунд, заданных параметром *ms*.

Допускается на вход и на выход подавать одну и ту же временную метку.

Объявлено в TMCPC.h

4.5.2.2. TMCPC_TS_CheckTime

Проверка корректности временной метки.

```
TMCP_ULONG TMCPC_TS_CheckTime(
   const S_TMCP_TimeStamp *ts
   );
```

- *ts* - Проверяемая временная метка.

Возвращаемое значение: 1 - нет ошибок, 0 - ошибка.

Объявлено в TMCPC.h

4.5.2.3. TMCPC_TS_CompareTime

Сравнение временных меток.

```
TMCP_LONG TMCPC_CompareTime(
   const S_TMCP_TimeStamp *ts1,
   const S_TMCP_TimeStamp *ts2
   );
```

- *ts1* - Первая временная метка.
- *ts2* - Вторая временная метка.

Возвращаемое значение: 0 - *ts1* равно *ts2*, 1 - *ts1* больше *ts2*, -1 - *ts1* меньше *ts2*.

Объявлено в TMCPC.h

4.5.2.4. TMCPC_TS_DecTime

Уменьшение времени на одну миллисекунду.

```
unsigned char TMCPC_TS_DecTime(
   const S_TMCP_TimeStamp *ts,
   S_TMCP_TimeStamp *ts_res
   );
```

- *ts* - [in] уменьшаемое время.
- *ts_res* - [out] результат.

Возвращаемое значение: 1 - нет ошибок, 0 - ошибка.

Примечание:

Допускается на вход и на выход подавать одну и ту же временную метку.

Объявлено в TMCPC.h

4.5.2.5. TMCPC_TS_GetLocalTime

Получение текущего локального времени.

```
unsigned char TMCPC_TS_GetLocalTime(
   S_TMCP_TimeStamp *ts
   );
```

- *ts* - [out] текущее время.

Возвращаемое значение: 1 - нет ошибок, 0 - ошибка.

Примечание:

Функция получает текущее локальное время, установленное на УС.

Объявлено в TMCPC.h

4.5.2.6. TMCPC_TS_GetMaxTime

Получение максимального времени.

```
unsigned char TMCPC_TS_GetMaxTime(
   S_TMCP_TimeStamp *ts
   );
```

- *ts* - [out] Получаемое максимальное время.

Возвращаемое значение: 1 - нет ошибок, 0 - ошибка.

Объявлено в TMCPC.h

4.5.2.7. TMCPC_TS_GetMinTime

Получение минимального времени.

```
unsigned char TMCPC_TS_GetMinTime(
   S_TMCP_TimeStamp *ts
   );
```

- *ts* - [out] Получаемое минимальное время.

Возвращаемое значение: 1 - нет ошибок, 0 - ошибка.

Объявлено в TMCPC.h

4.5.2.8. TMCPC_TS_IncTime

Увеличение времени на одну миллисекунду.

```
unsigned char TMCPC_TS_IncTime(
  const S_TMCP_TimeStamp *ts,
  S_TMCP_TimeStamp *ts_res
  );
```

- *ts* - [in] увеличиваемое время.
- *ts_res* - [out] результат.

Возвращаемое значение: 1 - нет ошибок, 0 - ошибка.

Примечание:

Допускается на вход и на выход подавать одну и ту же временную метку.

Объявлено в TMCPC.h

4.6. Прототипы функций обратного вызова

4.6.1. TMCPC_CONNECTION_HANDLER

Функция-обработчик соединения/разъединения.

```
typedef void (__stdcall *TMCPC_CONNECTION_HANDLER)(
  TMCPC_CON_REASON reason,
  U_TMCP_IPAddress addr,
  void *param
  );
```

- *reason* - идентификатор причины вызова функции.
- *addr* - если причина вызова - установка соединения, то параметр содержит IP адрес устройства TM.
- *param* - передаваемый в функцию параметр.

Примечание:

IP адрес формируется следующим образом: номера узлов записываются в 4-х байтовое поле в обратном порядке. Например, для адреса 10.7.1.179 (hex: A.7.1.B3) 4-байтовое значение будет: B301070Ah (3003189002).

Передаваемый в функцию параметр *param* можно задать в поле *connect_handler_param* структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Объявлено в TMCPC.h

4.6.2. TMCPC_NOTIFICATION_HANDLER

Функция-обработчик уведомлений.

```
typedef void (__stdcall *TMCPC_NOTIFICATION_HANDLER)(
  TMCP_NOTIFY notif,
  void *data,
  void *param
  );
```

- *notif* - идентификатор уведомления.
- *data* - данные, связанные с уведомлением.
- *param* - передаваемый в функцию параметр.

Примечание:

Передаваемые данные (*data*), связанные с уведомлениями:

Уведомление	**Содержимое pvData**
TMCP_NOTIF_VEHICLE	S_TMCP_VehNotif
TMCP_NOTIF_INCIDENT	S_TMCP_IncNotif
TMCP_NOTIF_STATISTICS	S_TMCP_TimeStamp

Передаваемый в функцию параметр *param* можно задать в поле *notify_handler_param* структуры S_TMCPC_ConParams (см. TMCPC_Connect).

Пример обработки уведомлений:

```
  #include <stdio.h>
#include <string.h>
#include "tmcpc.h"

extern TMCPC_HANDLE Connect(
        const S_TMCPC_ConParams *con_params,
        const char *password);

static void __stdcall NotificationHandler(
        TMCP_NOTIFICATION notif,
        void *data,
        void *) {
    switch(Notify) {
    case TMCP_NOTIF_VEHICLE: {
            S_TMCP_VehNotif *veh = (S_TMCP_VehNotif*)data;
```

```
            printf("Vehicle is detected!\n"
                   "\tlane: %u\n\tclass: %u\n\tspeed: %u\n",
                   veh->lane, veh->vehicle.vehicle,
                   veh->vehicle.speed
            );
            break;
        }
    case TMCP_NOTIF_INCIDENT: {
            S_TMCP_IncNotif *inc = (S_TMCP_IncNotif*)data;
            printf("Incident is detected!\n"
                   "\tlane: %u\n\tincident: %u\n",
                   inc->lane, inc->incident.incident
            );
            break;
        }
    case TMCP_NOTIF_STATISTICS: {
            S_TMCP_TimeStamp *ts = (S_TMCP_TimeStamp*)data;
            printf("Statistics is ready!\n"
                   "\t%u-%02u-%02u %02u:%02u:%02u:%03u",
                   ts->year + 2000,  ts->month, ts->day,
                   ts->hour, ts->min, ts->sec, ts->msec);
        break;
        }
    }
}

int main() {
    S_TMCPC_ConParams con_params = {0};
    TMCPC_HANDLE tmcpc = 0;

    strcpy(con_params.addr, "10.7.8.25");
    con_params.port_num = 52224;
    con_params.request_timeout = 10;
    con_params.notification_handler = NotificationHandler;

    tmcpc = Connect(&con_params, "tm");

        // Do anything...

    TMCPC_Disconnect(tmcpc);
    TMCPC_Release(tmcpc);
    TMCPC_Destroy(tmcpc);

    return 0;
}
```

Объявлено в TMCPC.h

НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

НТЦ Модуль
А/Я 166, Москва, 125190, Россия
Тел: +7 (499) 152-9698
Факс: +7 (499) 152-4661
E-Mail: rusales@module.ru
WWW: http://www.module.ru

Напечатано в России. Дата публикации: 16/06/2014