Описание протокола взаимодействия BLE Timecup CM-700

- Прием и передача данных ведется с использованием беспроводной технологии Bluetooth с низким энергопотреблением (BLE) версии не ниже 4.0 в режиме master-slave. Кофеварка является ведомым устройством (slave).
- Ведущее устройство передает команды ведомому путем пересылки командного пакета и следующей за ним контрольной суммы из двух байт (общая длина посылки - 20 байт).

- Структура командного пакета:

```
typedef struct _command_packet
{
unsigned char command;
unsigned char current_temperature;
unsigned char processor_temperature;
unsigned char heater_temperature;
unsigned char software_version;
unsigned char LEDs;  // не используется
unsigned char water_level;
// Нумерация рецептов начиная с нуля, сначала четыре пользовательских
// затем традиционный, быстрый, молоко.
unsigned char Recipe_number;                // номер рецепта
unsigned char target_temperature;           // целевая температура
unsigned char step_number;                  // номер шага в рецепте, начиная с нуля
unsigned char reserved_0;
unsigned char reserved_1;
unsigned char reserved_2;
char Coffee_Temperature_Correction;         // коррекция датчика температуры
unsigned short int Coffee_Cups_Number;      // количество завершенных рецептов
unsigned short int water_sensor_raw_data;   // сырые данные датчика пенки
} __command_packet;

#define COMMAND_PACKET_LENGTH 18
#define CRC_LENGTH 2
```

- Значения команд (значения поля command структуры `__command_packet`)

```
#define NO_COMMAND                    0  // не используется
#define LIQUID_BUTTON_PRESS           2  // Вызов нажатия клавиши «Выбор рецепта»
#define TIME_BUTTON_PRESS             3  // Вызов нажатия клавиши «Время/Температура»
#define PLUS_BUTTON_PRESS             4  // Вызов нажатия клавиши «+»
#define MINUS_BUTTON_PRESS            5  // Вызов нажатия клавиши «-»
#define START_BUTTON_PRESS            6  // Вызов нажатия клавиши «Start»
#define SET_LIQUID_TYPE               8  // не используется
#define SET_EDIT_MODE                10  // не используется
#define SET_HEATER_MODE              12 // не используется
#define READ_CUSTOM_RECIPES_DATA     17 // читать пользовательские рецепты из устройства
#define WRITE_CUSTOM_RECIPES_DATA    18 // записать пользовательские рецепты в устройство
#define READ_TRADITIONAL_RECIPE_DATA 19 // читать настройки традиционного рецепта из устройства
#define WRITE_TRADITIONAL_RECIPE_DATA 20 // записать настройки традиционного рецепта в устройство
#define RESET_TRADITIONAL_RECIPE_DATA 21 // установить стандартные настройки традиционного рецепта
#define READ_STATUS                  23 // читать статус устройства
#define RESET_CONTROLLER             24 // перезагрузка контроллера устройства
#define RUN_RECIPE                   25 // запустить рецепт на выполнение
#define STOP_RECIPE                  26 // остановить рецепт
#define BAD_DATA_RESULT              27 // используется только в посылке от slave
#define RESET_CUSTOM_RECIPE_DATA     28 // установить стандартные пользовательские рецепты
```

- Контрольная сумма вычисляется следующей функцией:

```
unsigned short crc16(volatile unsigned char *sbuf,unsigned short len)
{
  unsigned short crc=0xFFFF;

        while(len)
       {
              crc=(unsigned char)(crc >> 8) | (crc << 8);
              crc^=(unsigned char) *sbuf;
              crc^=(unsigned char)(crc & 0xff) >> 4;
              crc^=(crc << 8) << 4;
              crc^=((crc & 0xff) << 4) << 1;
              len--;
              sbuf++;
       }
    return crc;
}
```

- Структура настроек традиционного рецепта

```
typedef struct _recipe_settings
{
unsigned short int Coffee_Target_Temperature;        // 3 этап - температура приготовления
unsigned short int Coffee_Low_Power_level;           // 2 этап - мощность
unsigned short int Coffee_Middle_Power_level;        // 4 этап - мощность
unsigned short int Coffee_Impulses_Number;           // 5 этап - количество импульсов нагревателя
unsigned short int Coffee_Boiling_Threshold;         // Чувствительность датчика подъема пенки
unsigned short int Coffee_Initial_Power_Level;       // 1 этап - мощность
char Coffee_Temperature_Correction;                  // коррекция датчика температуры ****
char Coffee_BT_enable;                               // 1, если разрешено BLE  ****
unsigned short int Coffee_Cups_Number;               // количество завершенных рецептов  ****
// переменные времени устанавливаются по формуле (значение переменной) = (время в секундах)*64
// например, если нужна пауза в 10 сек, то переменная Coffee_Impulses_Pause = 640
unsigned long int Coffee_Impulses_Pause;             // 5 этап - пауза между импульсами нагревателя
unsigned long int Coffee_Impulses_Duration;          // 5 этап - длительность импульсов нагревателя
unsigned long int Coffee_Holding_Time;               // 3 этап - время приготовления
} __recipe_settings;

#define RECIPE_SETTINGS_LENGTH 28
```

***** Для поддержания этих настроек в актуальном состоянии перед записью новых значений структуры настроек следует предварительно считать эти значения из устройства, например, командой READ_STATUS.

- Структура данных пользовательских рецептов

```
#define  NUMBER_OF_STEPS_IN_RECIPE    28
#define  NUMBER_OF_RECIPES             4
#define  RECIPE_NAME_LENGTH           16
#define  RECIPE_ELEMENT_LENGTH        8
#define  RECIPES_DATA_LENGTH
((RECIPE_NAME_LENGTH+RECIPE_ELEMENT_LENGTH*NUMBER_OF_STEPS_IN_RECIPE)*NUMBER_OF_RECIPES)

typedef struct Recipe_Element
{
  unsigned short int Temperature;
  unsigned short int Power;
// переменные времени устанавливаются по формуле (значение переменной) = (время в секундах)*64
// например, если нужен отрезок времени в 10 сек, то переменная Time = 640
  unsigned long int Time;
} _Recipe_Element;

typedef struct _Recipe
{
  unsigned char name[RECIPE_NAME_LENGTH];
  _Recipe_Element Step[NUMBER_OF_STEPS_IN_RECIPE];
} __Recipe;

typedef union _custom_recipe_charint
{
    __Recipe current[NUMBER_OF_RECIPES];
    unsigned char char_custom_recipe[RECIPES_DATA_LENGTH];
} __custom_recipe_charint;
```

- В зависимости от команды, ведомое устройство действует по трем сценариям:

    1. Если команда входит в следующее множество:

```
#define NO_COMMAND                      0
#define LIQUID_BUTTON_PRESS             2
#define TIME_BUTTON_PRESS               3
#define PLUS_BUTTON_PRESS               4
#define MINUS_BUTTON_PRESS              5
#define START_BUTTON_PRESS              6
#define SET_LIQUID_TYPE                 8
#define SET_EDIT_MODE                   10
#define SET_HEATER_MODE                 12
#define RESET_TRADITIONAL_RECIPE_DATA   21
#define READ_STATUS                     23
#define RESET_CONTROLLER                24
#define RUN_RECIPE                      25
#define STOP_RECIPE                     26
#define BAD_DATA_RESULT                 27
#define RESET_CUSTOM_RECIPE_DATA        28
```

       ,то устройство выполняет команду и в ответ пересылает командный пакет `__command_packet` с полями, заполненными актуальной информацией и его контрольную сумму (общая длина ответной посылки - 20 байт).

    2. Если команда - `READ_TRADITIONAL_RECIPE_DATA` ,
    то ведомое устройство в ответ пересылает структуру `__recipe_settings` с полями, заполненными актуальной информацией и его контрольную сумму (общая длина ответной посылки - 30 байт).

    3. Если команда - `READ_CUSTOM_RECIPES_DATA` ,
    то ведомое устройство в ответ пересылает структуру `__custom_recipe_charint` с полями, заполненными актуальной информацией и его контрольную сумму (общая длина ответной посылки - 962 байт).

    4. Если команда - `WRITE_TRADITIONAL_RECIPE_DATA`,
    то ведомое устройство после приема командного пакета входит в режим приема структуры `__recipe_settings` и контрольной суммы (общая длина посылки - 30 байт), далее ведущее устройство после задержки около 100мс должно передать эту структуру, после чего ведомое устройство передает командный пакет `__command_packet` с полями, заполненными актуальной информацией и его контрольную сумму (общая длина ответной посылки - 20 байт)

    5. Если команда - `WRITE_CUSTOM_RECIPES_DATA` ,
    то ведомое устройство после приема командного пакета входит в режим приема структуры `__custom_recipe_charint` и контрольной суммы (общая длина посылки - 962 байт), далее ведущее устройство после задержки около 100мс должно передать эту структуру, после чего ведомое устройство передает командный пакет `__command_packet` с полями, заполненными актуальной информацией и его контрольную сумму (общая длина ответной посылки - 20 байт).