Конструктор спутника "ОрбиКрафт"

Принципы обмена сообщениями

В шине конструктора инициатором общения по шине может быть только бортовой компьютер управления. Он может опрашивать различные устройства в шине, в том числе и Arduino, в моменты времени, заданные пользователем. Также он может получать ответы, но только в течение ограниченного промежутка времени, которое пользователь может назначать при использовании описанной ниже функции. Поэтому каждый раз, когда мы хотим инициировать передачу или получение команд на Arduino, команда на такую операцию должна быть прописана в программе бортового компьютера.

Со стороны бортового компьютера

Команда на общение с Arduino подается при помощи следующей функции:

```
int arduino_send(const uint16_t num, const uint16_t msg_id, char *args, char *answer, char *timeout)
```

- Первый аргумент - номер Arduino (по умолчанию 0)
- Второй аргумент - идентификатор сообщения (см. какой необходим на стороне Arduino в зависимости от исполняемой программы)
- Третий аргумент - передаваемые данные (если ничего не передаем, то NULL)
- Четвертый аргумент - буфер для получаемых данных (если ничего не собираемся получать, то NULL)
- Пятый аргумент - время ожидания ответа от Arduino (в мс). В зависимости от решаемой задачи, данное значение должно выбираться пользователем.

Со стороны Arduino

На стороне Arduino используется специальная библиотека *orbicraftBus.arduinoLib.zip*, которая содержит различные методы для работы с бортовой шиной конструтора.

Описание методов:

Функция приема сообщения
При наличии нового сообщения оно записывается в переменную msg. Функция возвращает длину сообщения в байтах (включая идентификаторы), при отсутствии сообщения возвращает 0. При ошибке декодирования возвращает -1.

```
int16_t takeMessage(Message &msg)
```

Структура «Message» имеет следующие поля:

- from – идентификатор отправителя
- to – идентификатор получателя
- id – идентификатор сообщения
- data – строка аргументов

Функция отправки сообщения
Возвращает длину переданного сообщения в байтах.

```
int16_t sendMessage(const uint16_t address, const uint16_t id, const String data)
```

- address – адрес получателя

- id – идентификатор сообщения
- data – строка аргументов

Функция записи принятых данных в буфер
Данный метод должен вызываться из функции serialEvent2() - см. пример кода.

```
void serialEventProcess(void)
```

Функция установки номера устройства
Нужна в том случае, если на шине несколько Arduino.

```
void setArduinoNumber(const uint8_t newNumber)
```

newNumber — новый номер устройства.

Функция, возвращающая номер устройства

```
uint8_t getArduinoNumber(void)
```

Чтобы понять как все работает во взаимодействии конструктора ОрбиКрафт и Arduino, вам нужно изучить протокол передачи данных COBS [https://en.wikipedia.org/wiki/Consistent_Overhead_Byte_Stuffing] и основы объектно ориентированного программирования, термины из которого вы встретите в пояснениях к коду. Мы очень рекомендуем это сделать для понимания принципов работы информационной шины конструктора и ее взаимодействия с полезной нагрузкой. Также для написания кода вам понадобятся базовые знания языка программирования C/C++.

Если в данный момент у вас нет времени вникать в эти вещи, то вы можете продолжить работу просто используя готовые программы, либо создавая новые по образу и подобию предоставленных примеров.

Передача команд на Arduino

Если вы еще этого не сделали, то ознакомьтесь прежде с разделом подготовка к работе [http://orbicraft.sputnix.ru/doku.php?id=arduino_module_setting], где рассказывается о необходимых предварительных действиях.

По традиции при работе с Arduino, нашим первым примером будет мигание встроенным светодиодом. На шилде он отмечен как DIR_LED. При этом можно заметить, что на самой плате Arduino также зажигается встроенный светодиод, подключенный к пину 13.

Здесь мы используем следующий подход к использованию функции *arduino_send()*. Будем передавать только индентификатор сообщения от бортового комьютера, а на стороне Arduino по этому идентификатору будет просто вызываться готовая функция, написанная традиционным для среды Arduino способом.

Код для бортового компьютера

Здесь нашей задачей является передать сообщение с идентификатором, по которому будет выполняться функция на Arduino.

Можете вспомнить как правильно создавать программы для бортового компьютера в разделе Первое знакомство [http://orbicraft.sputnix.ru/doku.php?id=first_program].

ArduinoTestLed_1_Orbi.c

```
#include "libschsat.h"

void control(void)
```

```
{

        /*
        * Передадим команду на выполнение команды с идентификатором 1 на Arduino, используя встр
        * Первый аргумент - номер Arduino (по умолчанию 0)
        * Второй аргумент - идентификатор сообщения (в данном случае 1)
        * Третий аргумент - передаваемые данные (в данном случае NULL - данных для передачи нет)
        * Четвертый аргумент - буфер для получаемых данных  (в данном случае NULL - ответ не жде
        * Пятый аргумент - время ожидания ответа от Arduino (в данном случае 100 мс, но по сути
        */


        arduino_send(0, 1, NULL, NULL, 100);

}
```

Код для Arduino

Нашей задачей является включить по команде с бортового компьютера [http://orbicraft.sputnix.ru/doku.php?id=pc_subsys] встроенный светодиод на 3 секунды и выключить его. Для этого мы должны получить посылку с идентификатором от бортового компьютера и выполнить соответствующую команду - вызвать функцию по работе со светодиодом.

Не забудьте предварительно установить [https://www.arduino.cc/en/guide/libraries] в Arduino IDE библиотеку для взаимодействия Arduino и бортового компьютера. Библиотека находится в разделе необходимое ПО [http://orbicraft.sputnix.ru/doku.php?id=software].

ArduinoTestLed_1_Ard.ino

```
#include <OrbicraftBus.h> // подключаем библиотеку для работы с конструктором ОрбиКрафт

/*
  * Объявим переменную msg как тип данных Message
  * Message - представляет собой структуру, содержащую идентификаторы и данные передаваемого соо
*/
Message msg;

/*
  * Объявим переменную bus как тип данных OrbicraftBus
  * OrbicraftBus - представляет собой класс, описывающий взаимодействие Arduino и шины конструкт
*/
OrbicraftBus bus;

// Объявим переменную msgSize, в которую будет записываться размер принятого сообщения
uint16_t msgSize = 0;

void setup() {
  Serial2.begin(9600); // задаем скорость обмена информацией по Serial2 (протестируйте скорость
}

void loop() {
  msgSize = bus.takeMessage(msg); // пробуем прочитать сообщение с помощью метода takeMessage
  if (msgSize > 0){ //если сообщение есть
    switch (msg.id){//в зависимости от идентификатора сообщения выполняем те или иные действия

      // Рассмотрим случай с идентификатором 1
        case 0x01:
          turnOnLed(); // Вызов функции для включения и выключения светодиода
          break;
    }
  }
}
```

```
void turnOnLed(void){
  digitalWrite(LED_BUILTIN, HIGH); //Включаем встроенный светодиод
  delay(3000); //Ждем 3 секунды
  digitalWrite (LED_BUILTIN, LOW); //Выключаем встроенный светодиод
}

/*
 * Следующий блок кода необходимо всегда добавлять в конец программы
 * Функция вызывается автоматически и необходима для обработки сообщения
*/
void serialEvent2() {
  bus.serialEventProcess();
}
```

Получение данных от Arduino

Получение данных от Arduino к БКУ осуществляется с помощью той же функции на стороне БКУ:

```
int arduino_send(const uint16_t num, const uint16_t msg_id, char *args, char *answer)
```

Суть работы та же, что и при передаче команда с БКУ на Arduino. Используются для работы лишь другие аргументы функции *arduino_send* и соответственно нужно еще сформировать данные для передачи на стороне Arduino. Для примера будем принимать значения с фоторезистора (датчик освещенности), подключенного к пину **A0** Arduino. Примеры кода с пояснениями представлены ниже.

Прежде чем подключать фоторезистор к Arduino, посмотрите как это правильно делается. Фоторезистор может быть использован в виде отдельного элемента с необходимой периферией, а может быть в виде готового модуля с готовой обвязкой.

Код для бортового компьютера

ArduinoTestLightSensor_1_Orbi.c

```
#include "libschsat.h"

        void control(void)
        {
                char answer[255]; // Создаем массив для сохранения ответа
                int32_t count = 5; // Устанавливаем счетчик на 5 шагов
/*
        * Передадим команду на получение ответа с идентификатором 2 на Arduino, используя встрое
        * Первый аргумент - номер Arduino (по умолчанию 0)
        * Второй аргумент - идентификатор сообщения (в данном случае 2)
        * Третий аргумент - передаваемые данные (в данном случае NULL - данных для передачи нет)
        * Четвертый аргумент - буфер для получаемых данных  (в данном случае answer - массив для
        * Пятый аргумент - время ожидания ответа от Arduino в мс (в данном случае 100 мс)
        */
                while (count > 0){
                        int status = arduino_send(0, 2, NULL, answer, 100);
                        if (status == 0){
                                printf("Answer: %s\r\n", answer);
                        }
                        else{
                                printf("Error\r\n");
                        }
                        mSleep(500);
                        count--;
                }

        }
```

Код для Arduino

ArduinoLightSensor_1_Ard.ino

```
#include <OrbicraftBus.h>

/*
  * Объявим переменную msg как тип данных Message
  * Message - представляет собой структуру, описывающую идентификаторы передаваемого сообщения
*/
Message msg;

/*
  * Объявим переменную bus как тип данных OrbicraftBus
  * OrbicraftBus - представляет собой класс, описывающий взаимодействие Arduino и шины конструкт
*/
OrbicraftBus bus;

// Объявим переменную msgSize, в которую будет записываться передаваемое сообщение
int16_t msgSize = 0;
// Объявим номер пина для считывания показаний
int data_pin = A0; // Указываем пин, с которого будем считывать показания датчика

void setup() {
  Serial1.begin(9600); // задаем скорость обмена информацией по Serial2 (протестируйте скорость
}

void loop() {

  msgSize = bus.takeMessage(msg); // пробуем прочитать сообщение с помощью метода takeMessage

  if (msgSize > 0){ //если сообщение есть
    switch (msg.id){//в зависимости от идентификатора сообщения выполняем те или иные действия

      // Рассмотрим случай с идентификатором 2
        case 0x02:{
        String data = String(Sensor_data()); // записываем показания, полученные от функции Sens
        bus.sendMessage(bus.obcAddress, 0, data); // передаем содержимое переменной data на БКУ
        break;
      }
    }
  }
}

uint16_t Sensor_data(void){
  uint16_t data = analogRead(data_pin); //Считываем показания освещенности с датчика
  return data;
}

/*
 * Следующий блок кода необходимо всегда добавлять в конец программы
 * Функция вызывается автоматически и необходима для обработки сообщения
*/
void serialEvent2() {
  bus.serialEventProcess();
}
```