Лисий медальон

# Общее описание

Лисьи медальоны должны чуять друг друга за 30 м. Величина чувствительности задается байтом в EEPROM, меняется по радио.

При появлении в радиусе одного или нескольких других медальонов медальон начинает вздрагивать: 1...3 медальона – соответственно 1...3 раза в секунду; больше трех – четыре раза в секунду.

# Протокол обмена данными

У каждого медальона есть уникальный адрес. Этот адрес хранится в EEPROM и на момент прошивки равен 1.

Если медальон А принимает пакет от медальона Б, адрес коего совпадает с адресом А, то А увеличивает свой адрес на 1 и записывает его в EEPROM.

Принятые пакеты с уровнем сигнала менее порогового игнорируются.

В дальнейшем длительность пакета обозначается tp. Другой – передающий – медальон обозначается МА; медальон, о коем идет речь, обозначается МБ.

Время делится на циклы. Длительность одного цикла – 500 мс.

МБ слушает эфир, пока не его очередь передавать.

При получении чужого пакета:

* Если адрес МА равен адресу МБ, то МБ модифицирует свой адрес и снова начинает цикл;
* Если адрес МА меньше адреса МБ, то МБ синхронизирует по МА свои часы.
* Если адрес МА больше адреса МБ, то МБ ничего не меняет.

МБ должен передавать пакет независимо от того, сколько других медальонов он видит. Это нужно для обеспечения корректной работы тех медальонов, кои не слышат других.

# Список команд

|  |  |  |
| --- | --- | --- |
| **Название** | **Код** | **Данные** |
| Call | 0xCA | 0x11 – первый байт, свой адрес – второй байт, свое время (TCNT1) – третий и четвертый байты **TODO**. Broadcast. «Я тут!» |
| Acknowledge | 0xAC | Подтверждение. Первый байт – свой адрес, второй – уровень принятого сигнала |
| SetSensitivity | 0x55 | Первый байт – уровень задаваемой чувствительности. Повторять до получения ACK. |
| Setup channel | 000 | Два байта задают номер канала. Нужно при переходе на другую частоту. Повторять до получения ACK. |

# Путь прогресса

На очередном этапе я получил переполнение RX-буфера. Очевидно, пришел новый пакет, пока старый обрабатывался, или еще что.

Поэтому применен другой механизм смены состояний.

* На старте входим в RX.
* При приеме пакета СС переходит в IDLE режим.
* СС загоняется обратно в RX по завершению опустошения входного буфера.
* При наступлении времени передачи заполняется TX буфер, выставляется флаг TX\_NEEDED.
* В СС-таске проверяется, нет ли несущей. Если нет, входим в TX режим.
* После завершения передачи СС автоматически переходит в RX.

Вилы: при некорректном приеме пакета СС переходит в IDLE, при этом не дергая прерывание. Логично, в общем-то. Поэтому режим RX->IDLE, из коего можно выйти только по прерыванию, неправилен.

Значит, таки нужен RX\_NEEDED.

Далее.

Всё сделано на RX\_NEEDED – TX\_NEEDED. Без передачи всё принимается – или не принимается – нормально. Если же включить передатчик, то всё принимается/передается, пока не появится чужой сигнал. Тогда State=17 (RX FIFO overflow), и мы зависаем. Интересно, где, кстати.

Проверил. Нет, я всё понимаю. Но: пакеты-то в СС шлются. И ответы получаются. То есть, вовсе мы не висим. Но почему тогда ничего не говорится в УАРТ?

Убрал проверку получения пакета. Получил несколько overflow, потом опять зависли.

Слава красному диоду. Зависли в CC\_TASK.

Чудесно. Закомментил ENTER\_TX, всё, порядок.

Итак, виснем в момент записи строба ENTER\_RX. В состоянии переполнения.

Эге. Очень похоже, что это тупо из-за прерывания в этот славный момент While();

Так и есть. Какой я молодец, а. Люди, будьте бдительны с прерываниями.

Тем временем, мы словили переполнение и не зависли. Ура. Теперь нужно что-то с этим сделать.

Добавил обработку состояний – при переполнении или наоборот. Всё теперь стабильно.

Так. Ура, мы друг друга *вроде* видим. Вроде, поелику №1 горит лампочкой, а №2 – хорошо если мигает. Но №2 – без антенны. Видимо, сие не мешает ему передавать, а вот принимать... Хм. Подозрительно.