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

Адрес любого из медальонов – 1. В дальнейшем медальон может подписываться и другим адресом, но адрес в СС это не изменяет. Поскольку незачем. А динамически измененный адрес всё равно непросто использовать.

То есть, адрес «FROM» динамичен, адрес «TO» ниезменен и равен 1.

Если медальон А принимает пакет от медальона Б, адрес коего совпадает с адресом А, то А увеличивает свой адрес на 1. После этого А слушает еще один цикл без передачи.

Принятые пакеты с уровнем сигнала менее порогового игнорируются.

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

Время делится на циклы. Длительность одного цикла – 200 мс. Каждый пятый цикл М включается на прием и делает всё, что должно. Остальное время М не принимает, а только передает в положенное время. М должен передавать пакет независимо от того, сколько других медальонов он видит. Это нужно для обеспечения корректной работы тех медальонов, кои не слышат других.

О приеме (цикл 0).

МБ слушает эфир, пока не его очередь передавать. При получении чужого пакета:

* Если адрес МА равен адресу МБ, то МБ модифицирует свой адрес и подстраивает по МА свои часы;
* Если адрес МА меньше адреса МБ, то МБ подстраивает по МА свои часы и передает в свое время;
* Если адрес МА больше адреса МБ, то МБ ничего не меняет и передает в свое время.

# Список команд

|  |  |  |
| --- | --- | --- |
| **Название** | **Код** | **Данные** |
| Call | 0xCA | 0x11 – первый байт, свой адрес – второй байт, свое время (TCNT1H, L) – третий и четвертый байты. 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. |

# Knowledge base

Следует программировать фьюз EESAVE – чтобы адрес не терялся при следующих перепрошивках.

WOR – это просто таймер, встроенный в СС. ПО нему она может в заданный промежуток времени просыпаться, заданное время слушать канал, и засыпать обратно. Нам это не очень ценно, поскольку нужно слушать идин цикл, а остальное время периодически передавать.

# TODO

~~Watchdog.~~

Слушаем 1/5 секунды, передаем циклично.

~~Перейти на status byte.~~

# 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.

*13.04.10*

Модифицировать проверку наличия несущей перед передачей. СС всё равно не войдет в режим передачи, пока всё не примет; но факт текущего приема пакета означает необходимость пересобрать свой. Итого, если нужно передавать, а мы принимаем – отменить передачу, очистить TX буфер.

Похоже, переполнение TX-буфера называется TX\_UNDERFLOW.

Перейти на статус-байт, избавившись от этой тучи режимов.

*Later*

Итак, есть работоспособный вариант на фиксированных адресах. Адреса читаются из EEPROM. И всё очень мило синхронизируется. Расстояние между передачей первого и второго – 25...27 мс. При том что таймслот получается путем умножения адреса на 16. То есть, в идеале должно было бы быть 16 мс, но еще добавляется сколько-то задержек (скорее всего, от UART).

На статус-байт не перешли. Возможно, пока, а может, и вовсе не перейдем. Ибо неясно, зачем.

Так, теперь нужно: (1) проверить с тремя медальонами и (2) вернуться к динамическим адресам.

Еще нужно уменьшить множитель, поскольку 27 мс – это многовато. В 200 мс 10 медальонов не влезут. Тем более что CarrierDetect говорит, что длина передачи 6 мс. Но первым делом – отключить UART.

*14.04.10*

Убрал всё про УАРТ. Расстояние меж пакетами не изменилось, стабильность смещения тоже не улучшилась.

Сменил множитель с 16 на 8. Расстояние уменьшилось до 17.8 мс. Меньше плавает.

Сменил множитель с 8 на 4. Расстояние уменьшилось до 13.4 мс. Принятый пакет поступает через 7.4 мс после прерывания на передачу. Длительность пакета примерно 6.8 мс – измерено по CarrierDetect.

Так. Адрес 1 означает 4 попугая, адрес 2 – 8 попугаев. Длина одного попугая – примерно 1мс. Итого, будь всё честно, расстояние между пакетами было бы 4 мс.

То есть, у нас вносится задержка в 10 мс. То есть, роль таймера становится довольно убогой. Почему – понятно: после приема пакета мы выставляем текущее время таким, каким оно было в момент срабатывания прерывания на передачу у первого медальона. Не учитывая время подготовки пакета и его передачу.

Чем это плохо. Если медальон 10 примет пакет медальона 1, он захочет передавать в момент 40. А это неверно.

Попробуем учесть. Будем добавлять к таймеру 7. Таким образом, при множителе 4 мы упустим время передачи: 1\*4+7=11 > 8. При множителе 8: 1\*8+7 = 15 < 2\*8=16. Маловато времени, вообще-то.

При длине пакета 7 дребезг был в 2 мс. При 8 – почти исчез.

Итак, длина пакета близка к 8, 9 уже слишком много. Перейти, что ли, на другое деление частоты таймером? Перейду на 1/64ю.

*Later*

Время от момента прерывания на передачу до окончания приема пакета – 8.4 мс.

Перешел на быстрый таймер. Второй сигнал почти не дрожит. Множитель 256, длина пакета 131.

При этом множителе рассчетное расстояние между передачами 16.38 мс. Экспериментальное – 16.9 мс. Очевидно, 500 мкс ушло на какие-то инструкции и округления. Уменьшить множитель нельзя, так как тогда расстояние между передачами сократится до 8 с чем-то мс, а это меньше длины приема пакета. Впрочем, об этом мы уже писали.

Прибрался в файлах немного. Выкинул RX\_needed. Оно всегда needed.

Добавил динамические адреса. Всё работает. Подозрительно.

*15.04.10*

WOR – отказать. Проще и нагляднее сделать всё АВРом, чем подбирать параметры не предназначенного для наших целей функционала СС.

Надо, надо переходить на статус-байт. MarcState is mainly for test purposes.

*17.04.10*

Надо ли переходить? Мы сможем опознать переполнение?