Amomum 8 февраля 2019 в 12:50

RS-485 на отечественных микроконтроллерах от фирмы Миландр

Программирование микроконтроллеров, Электроника для начинающих

Несколько дней назад я имел неосторожность завуалированно пообещать запилить пост про Миландр… Ну что ж, попробуем.

Как вы, вероятно, уже знаете, существует российская компания Миландр, которая, среди прочего, выпускает микроконтроллеры на ядре ARM Cortex-M. Волею судеб я был вынужден с ними познакомиться достаточно плотно, и *познал боль*.

Небольшая часть этой боли, вызванная работой с RS-485, описана далее. Заранее прошу прощения, если слишком сильно разжевываю базовые понятия, но мне хотелось сделать эту статью доступной для понимания более широкой аудитории.
Так же заранее оговорюсь, что имел дело только с 1986BE91 и 1986BE1, о других уверенно говорить не могу.

TL; DR

Вступление

Интерфейс RS-485 (так же известный как EIA-485, хотя я ни разу не слышал, чтобы его так называли в обиходе) – это асинхронный полудуплексный интерфейс с топологией «шина». Этот стандарт оговаривает только физику — т.е. уровни напряжения и временные диаграммы — но не оговаривает протокол обмена, защиту от ошибок передачи, арбитраж и тому подобное.

По факту, RS-485 — это просто полудуплексный UART с повышенными уровнями напряжения по дифференциальной паре.
Именно эта простота и обеспечивает популярность RS-485.
Чтобы превратить UART в RS-485 используются специальные микросхемы-преобразователи, такие как MAX485 или 5559ИН10АУ (от того же Миландра). Они работают почти «прозрачно» для программиста, которому остается только правильно выбирать режим работы микросхемы – прием или передача. Делается это с помощью ног nRE (not Receiver Output Enable) и DE (Driver Output Enable), которые, как правило, объединяются и управляются одной ногой микроконтроллера.

Поднятие этой ноги переключает микросхему на передачу, а опускание — на прием.
Соответственно, все, что требуется от программиста, это поднять эту ногу RE-DE, передать нужное количество байт, опустить ногу и ждать ответа. Звучит достаточно просто, правда?
Хе-хе.

Проблема

Эту ногу нужно опустить в тот момент, когда все передаваемые байты полностью переданы на линию. Как поймать этот момент?
Для этого нужно отловить событие «Transmit complete» (передача завершена), которое генерирует блок UART'а в микроконтроллере. В большинстве своем события – это выставление бита в каком-нибудь регистре или запрос прерывания.
Чтобы отловить выставление бита в регистре, регистр нужно опрашивать, т.е. использовать код, вроде этого:

```
while( MDR_UART1->FR & UART_FR_BUSY ) {;}
```

Это если мы можем себе позволить полностью остановить выполнение программы, пока все байты не будут переданы. Как правило, мы себе этого позволить не можем.

Прерывание в этом отношении гораздо удобнее, поскольку оно прилетает само по себе, асинхронно. В прерывании мы можем быстренько опустить RE-DE и всего делов.

Разумеется, если бы мы могли так сделать, никакой боли бы не было и этого поста бы тоже не было.

Дело в том, что в блоке UART, который Миландр ставит во все свои микроконтроллеры на Cortex-M (насколько мне известно), нет прерывания по событию «Передача завершена». Есть только флаг. И есть прерывание «Буфер передатчика пуст». И прерывание «байт принят», конечно же.

Еще есть

Проблема в том, что «Буфер передатчика пуст» – это совсем не то же самое, что «Передача завершена». Насколько я понимаю внутреннее устройства UART'а, событие «Буфер пуст» означает, что в буфере передатчика есть хотя бы одно свободное место. Даже в случае, если это место всего одно (т.е. буфер размером в один байт), это лишь означает, что последний передаваемый байт был скопирован во внутренний сдвиговый регистр, из которого этот байт будет выползать на линию, бит за битом.

Короче говоря, событие «буфер передатчика пуст», не означает, что все байты были переданы полностью. Если мы опустим RE-DE в этот момент, то мы «обрежем» нашу посылку.

Что же делать?

Ребус



Расшифровка:

«Прополка битовых полей» — это локальный мем из короткой, но наполненной болью темы на форуме Миландра — forum.milandr.ru/viewtopic.php?f=33&t=626.
Простейшее решение – это таки «пропалывать» (от английского «poll» — непрерывный опрос) флаг UART_FR_BUSY.

Разумеется, это решение не очень приятно. Если мы не можем блокирующе проверять этот флаг, то приходится проверять его периодически. Чтобы проверять его периодически, приходится городить целый огород (особенно, если вы хотите написать переносимый модуль, а не просто разово решить эту проблему).

Если использовать какую-нибудь ОСРВ, то ради этой прополки приходится заводить целую отдельную задачу, будить ее в прерывании, ставить ей не самый низкий приоритет, морока, короче.

Но, казалось бы, ладно, помучились разок, дальше используем и радуемся. Но нет.
К сожалению, нам мало опустить RE-DE строго после того, как все байты были переданы до конца. Нам нужно опустить ее *не*

слишком поздно. Потому что мы на шине не одни. На наше сообщение, скорее всего, должен прийти какой-то ответ от другого абонента. И если мы опустим RE-DE слишком поздно, мы не переключимся в режим приема и потеряем несколько бит ответа.

Время, которое мы можем себе позволить «передержать» ногу RE-DE, зависит, в основном, от скорости передачи (бодрейта) и от быстроты устройства, с которым мы общаемся по шине.
В моем случае скорость была относительно невелика (57600 бод), а устройство было достаточно резвым. И иногда так случалось, что у ответа терялся бит-другой.

В целом, не очень хорошее решение.

Таймер

Второй вариант, который приходит в голову – использовать аппаратный таймер. Тогда в прерывании «Буфер передатчика пуст» мы запускаем таймер с таймаутом, который равен времени передачи одного байта (это время легко вычисляется из бодрейта), а в прерывании от таймера — опускать ногу.

Хороший, надежный способ. Только таймер жалко; их у Миландров традиционно немного – две-три штуки.

Режим шлейфа

Если внимательно читать тех. описание на UART — например, для 1986BE91T — можно заметить вот этот очень короткий абзац:

```
Проверка по шлейфу

Проверка по шлейфу (замыкание выхода передатчика на вход приемника) выполняется путем установки в 1 бита LBE в регистре управления контроллером UARTCR.
```

Если же тех. описание не читать, то практически того же эффекта можно добиться, закоротив ноги RX и TX аппаратно.

Мысли вслух

Идея состоит в следующем — перед передачей последнего байта в посылке, нужно активировать режим «проверки по шлейфу». Тогда можно получить прерывание по приему нашего собственного последнего байта в тот момент, когда он полностью вылезет на шину! Ну, почти.

На практике оказалось, что прерывание по приему срабатывает *немножко* раньше, чем должно, примерно на треть битового интервала. Я не знаю с чем это связано; возможно, в режиме проверки по шлейфу не происходит настоящего сэмплирования линии, может быть, режим шлейфа не учитывает последний стоп-бит. Не знаю. Как бы то ни было, мы не можем опустить RE-DE сразу по входу в это прерывание, потому что так мы «отрежем» от нашего последнего байта стоп-бит или часть стоп-бита.

Строго говоря, можем или не можем зависит от соотношения скорости работы интерфейса (т.е. длительности одного битового интервала) и частоты работы микроконтроллера, но я на 80 МГц тактовой частоты и с бодрейтом 57600 не мог.

Далее возможны варианты.

Если вы можете себе позволить опрашивать флаг UART_FR_BUSY в течение одного битового интервала — на деле даже чуть меньше, потому что вход в прерывание и предварительные проверки тоже отнимают время — то выход найден. Для скорости 57600 максимальное время опроса составит ~18 микросекунд (один битовый интервал), на практике — около 5 микросекунд.

Для тех, кому интересно, привожу код обработчика прерывания целиком.

```
void Handle :: irqHandler(void)
{
    UMBA_ASSERT( m_isInited );

    m_irqCounter++;

    // ------------------------------------------- Прием

    // do нужен только чтобы делать break
    do
    {
```

```
if ( UART_GetITStatusMasked( m_mdrUart, UART_IT_RX ) != SET )
    break;

// по-факту, прерывание сбрасывается при чтении байта, но это недокументированная фича
UART_ClearITPendingBit( m_mdrUart, UART_IT_RX );

uint8_t byte = UART_ReceiveData( m_mdrUart );

// для 485 используется режим шлейфа, поэтому мы можем принимать эхо самих себя
if( m_rs485Port != nullptr && m_echoBytesCounter > 0 )
{
    // эхо нам не нужно
    m_echoBytesCounter--;

    if( m_echoBytesCounter == 0 )
    {
        // после последнего байта надо __подождать__,
        // потому что мы принимаем его эхо до того, как стоп-бит до конца вылезет на линию
        // из-за мажоритарной логики семплирования.
        // Если не ждать, то можно потерять около трети стоп-бита.

        // Время ожидания зависит от бодрейта, примерное время ожидания:

        // бодрейт  / длительность бита,  /  время ожидания,  /
        //          /        мкс          /       мкс         /
        //          /                     /                   /
        // 9600     /      105            /       32          /
        // 57600    /       18            /       4,5         /
        // 921600   /        1            /        0          /
        //          /                     /                   /

        // при использовании двух стоп бит и/или бита четности,
        // время прополки вроде как не меняется.
        // Видимо, пропалывается только треть последнего бита, не важно какого.

        // блокирующе пропалываем бит
        while( m_mdrUart->FR & UART_FR_BUSY ) {;}

        // и только теперь можно выключать передатчик и режим шлейфа
        rs485TransmitDisable();

        // семафор, что передача завершена
        #ifdef UART_USE_FREERTOS
            osSemaphoreGiveFromISR( m_transmitCompleteSem, NULL );
        #endif
    }

    break;
}

// если в приемнике нет места - байт теряется и выставляется флаг overrun
#ifdef UART_USE_FREERTOS

    BaseType_t result = osQueueSendToBackFromISR( m_rxQueue, &byte, NULL );

    if( result == errQUEUE_FULL )
    {
        m_isRxOverrun = true;
    }

#else

    if( m_rxBuffer.isFull() )
    {
        m_isRxOverrun = true;
    }
    else
    {
        m_rxBuffer.writeHead(byte);
    }
```

```
        #endif


    } while( 0 );

    // ------------------------------------------- Ошибки

    // Проверяем на ошибки - обязательно после приема!
    // К сожалению, функций SPL для этого нет
    m_error = m_mdrUart->RSR_ECR;

    if( m_error != error_none )
    {
        // Ошибки в регистре сбрасывается
        m_mdrUart->RSR_ECR = 0;
    }

    // ------------------------------------------- Передача

    if( UART_GetITStatusMasked( m_mdrUart, UART_IT_TX ) != SET )
        return;

    // предпоследний байт в 485 - включаем режим шлейфа
    if( m_txCount == m_txMsgSize - 1 && m_rs485Port != nullptr )
    {
        setEchoModeState( true );
        m_echoBytesCounter = 2;
    }
    // все отправлено
    else if( m_txCount == m_txMsgSize )
    {
        // явный сброс можно (и нужно) делать только для последнего байта
        UART_ClearITPendingBit( m_mdrUart, UART_IT_TX );
        m_pTxBuf = nullptr;

        return;
    }

    // Еще есть, что отправить
    UMBA_ASSERT( m_pTxBuf != nullptr );

    UART_SendData( m_mdrUart, m_pTxBuf[ m_txCount ] );
    m_txCount++;
}
```

Если вы можете себе позволить перемычку (в идеале – управляемую) между ногами RX и TX, то всё тоже хорошо.

К сожалению, на сегодняшний день других вариантов я предложить не могу.

На этом у меня все. Если кому-нибудь известны другие способы решения этой проблемы, прошу поделиться ими в комментариях.

Так же, пользуясь случаем и изменением правил Хабра, хочу пропиарить сайт StartMilandr, который представляет собой собрание статей о микроконтроллерах Миландр. По неясной причине нагуглить его можно разве что случайно.

И, конечно же, напомнить о существовании форка стандартной периферийной библиотеки, в котором, в отличие от официальной библиотеки, исправляются баги и есть поддержка gcc.

Теги: миландр, 1986ве1, 1986ве91т, UART, RS-486, боль и унижение

Хабы: Программирование микроконтроллеров, Электроника для начинающих

@Amomum
Пользователь

ПОХОЖИЕ ПУБЛИКАЦИИ

15 февраля 2019 в 17:36
Удалённое управление UART'ом через Web
+37 9,7k 74 31

11 февраля 2019 в 10:36
Боль в спине — понимание с позиции современной медицины
+59 145k 223 231

19 октября 2018 в 15:29
Мой «Hello World!» на FPGA или очередная версия UART
+19 10,6k 71 36

Комментарии 61

Borjomy 8 февраля 2019 в 13:03 −2

Так надо понимать, что реакция Миландра на эти проблемы чисто совковая: вот вам изделие и разруливайте проблемы сами.

Amomum 8 февраля 2019 в 13:07 +8

Ну, их можно понять. Большая часть процессоров разработана для выполнения ОКР под конкретных заказчиков и их требования. Заказчики в основном военные и космос, им процы нужны под весьма специфические задачи. Эти задачи, судя по всему, процы выполняют.

Все остальное — пластиковые корпуса, форум, бесплатная техподдержка, отладочные комплекты, ПО — это полностью их инициатива, за которую им никто напрямую не платит. Под все это явно не хватает и денег и квалифицированных людей.

Borjomy 8 февраля 2019 в 13:31 +3

Думаю, отношение к оборонке точно такое-же. Аппаратно-то все соответствует. А все остальное — это уже проблемы программиста.

Amomum 8 февраля 2019 в 13:55 +7

Мы у них только готовые процы покупали, так что судить не берусь.
Тем не менее, по сравнению с другими отечественными микроконтроллерами, у Миландра все не так уж плохо — документацию можно скачать с сайта, а не запрашивать, отладочные комплекты можно купить (и даже не за космические деньги), на форуме отвечают хоть как-то, техподдержка тоже в наличии.

По идее, все это болезни роста.

el_kornholio 8 февраля 2019 в 14:43 +1

Полностью так. И приходится городить костыли (у нас было реализовано задержкой просто). Ну и еще — очень классно, когда вместо прерывания по фронту GPIO-input есть только по уровню.

Amomum 8 февраля 2019 в 14:57 +6

Если мы сейчас начнем все косяки перечилсять, то весь день потратим :)
Но почему нет?

Вкратце, что мы нашли:
— Уровней приоритета прерываний 8, а не 16, как у STM.
— В GPIO нельзя просто писать в регистр RXTX, если на этому порту висит JTAG.
— Датчик температуры — не датчик и не температуры, использовать его нельзя.
— Делители частоты АЦП на самом деле идут только до 2048.
— Чтобы использовать прерывание от DMA, нужно запретить прерывания от SSP.
— LSI можно выключить, а от него питается Watchdog.
— Функции для работы с flash-памятью (которую Миландр упорно называет EEPROM) должны располагаться в RAM.

Плюс к этому на 1986BE1:
— Нужно снимать галочку View->Periodic window update при использовании Кейла, иначе будет HardFault.
— Стирать flash из Кейла можно только целиком, посекторное стирание не поддерживается.
— Оператива делится на два куска, при этом код можно запускать только из второго, DMA может работать только со вторым и при прошивании можно использовать только второй кусок.
— SysTick считает медленнее, чем должен.
— Ethernet в SPL сломан, что смогли — пофиксили в форке. Стабильно работает только линейный режим.
— Доступ к внутреннему буферу Ethernet на запись должен быть кратен 4 байтам, если не кратен — то тихо записывается мусор. Соответственно, все счетчики и Delimiter должны быть всегда кратны 4, иначе будет черт знает что и внезапные HardFault'ы.

JerleShannara 9 февраля 2019 в 01:44 +3

Ну Ethernet хоть как-то работает. А то когда на вопрос «а как с вашие езернетом работать?», получаешь «а мы его сами не проверяли, если заставите — напишите нам как у вас это получилось», лицо несколько перекашивается и в голову лезут неприличные извращенные мысли с участием разработчиков камня.

Amomum 9 февраля 2019 в 02:10 +1

Ну как сказать — работает. В примере от Миландра, где все через регистры, волшебные числа и написано *очень странно* — да, работает :) А через их же библиотеку — не работает.

NORI 9 февраля 2019 в 17:33 0

Дело в том, что в Миландре почти всей разработкой занимаются только что закончившие ВУЗ студенты. Некоторые модули были взяты с минимальными переделками из предыдущих серий их PIC процессоров.
Причем все это в Зеленограде, где еще остались специалисты — разработчики.

Amomum 9 февраля 2019 в 17:42 +1

Звучит правдоподобно, конечно, хотя и бездоказательно.
Ходит еще шутка, что спецификации студенты за зачет пишут, слишком уж опечаток много. И из-под картинок иногда вылезали «оригиналы» на английском.

NORI 9 февраля 2019 в 20:00 0

Я говорил о сырости разработок и не собирался никому ничего доказывать. Потому что я оттуда.

Amomum 9 февраля 2019 в 22:27 0

Оу. Сочувствую.

progchip666 11 февраля 2019 в 12:46 0

Ну есть ещё старички — мой хороший знакомый предпенсионного возраста там трудится например. Зрелые разработчики в самом расцвете сил конечно там особо не задерживаются.

IgorPie 8 февраля 2019 в 15:47 +2

Если они делают по спецификации Cortex, то должно быть все, что указано в спецификации, CMSIS должен работать у всех производителей, хоть у Миландр, хоть у LPC, разве нет?
Иначе это нарушение спецификации и слово «кортекс» надо брать в кавычки, это обычный «Почин», где у них «Группен-секс», а мы им ответим нашим «бригаден-семером».

Amomum 8 февраля 2019 в 16:25 +6

Ааа, вот тут очень хитро :) Единственная ошибка, вроде бы относящаяся к ядру — это ошибка тактирования SysTick. Но она есть только у 1986BE1. А он, видите ли, не Cortex! Официально там «высокопроизводительное RISC-ядро», которое совершенно случайно практически идентично Cortex-M0. Абсолютно случайно.

arkamax 8 февраля 2019 в 18:43 +1

> совершенно случайно практически идентично Cortex-M0. Абсолютно случайно.

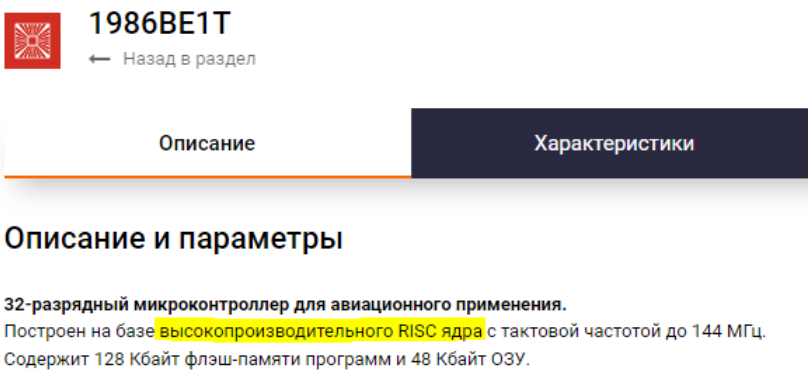
Очень напоминает одного моего китайского поставщика кабелей с проприетарными разъемами. Продают как то, что должно работать, по факту имеем такие допуски, что кабель приходится из эталонной «мамы» плоскогубцами вытаскивать. Но вообще совместим, да.

Demonter 10 февраля 2019 в 11:19 +2

Мне почему-то помнится, что в 1986ве1 ядро Cortex M1, которое предназначено для применения в плисинах. Оригинальный камень. В одном изделии использовали их для Ethernet стыка. Ради справедливости должен отметить, что по сравнению с теми же Элвисовскими камнями миландровские изделия исключительно user-friendly как с точки зрения документации, так и технической поддержки в форуме.

Amomum 10 февраля 2019 в 11:22 0

Нет-нет, что вы. Там «высокопроизводительное RISC ядро» :) Просто регистры совпадают с Cortex M1. И система команд. И в примерах почему-то есть core_cm1.h. Но это просто совпадение, разумеется.



progchip666 9 февраля 2019 в 10:35 +2

Задачи настолько специфические что не используют обмена по UART?
Уверен, что там и с SPI и с I2C косяков не меньше, только бороться ембеддерам с ними ещё сложнее.
Что касается миландра, то зачем им париться. Военные и ФСБшники всё равно купят, у них нет выбора, а ни один коммерческий проект в здравом уме и памяти всё равно их замечательные изделия использовать не будет по причине полной неконкурентоспособности.
Так что у них вполне разумная маркетинговая политика.

> ПО — это полностью их инициатива, за которую им никто напрямую не платит

— надо избавляться от такого подхода. Вы ещё напишите что пенсии нам государство платит. Нет, пенсия ~~которая теперь мне не светит~~ — заплаченные мной налоги. Я бы мог их не платить и сделать накопления на старость самостоятельно, но у меня нет альтернативы, приходится вместо этого финансировать Миландры.
Но мы ушли в сторону. TI ~~к примеру~~ вроде тоже никто не платит, за исходники, примеры и даже даташиты. Но они их делают, потому что их цель сделать полноценный продукт для конкуренции на глобальном рынке.
По этой причине мне в кошмарном сне не привидится разработать устройство на Миландре, ну а работающие на ВПК будут вынуждены и дальше тратить своё время и деньги не столько на программный продукт, сколько на костыли для обхода багов. Однако им же даёт их, и щедро даёт, государство. Полетит или не полетит ракета сегодня особо никого не волнует, всё равно поставят на вооружение — ядрёная то война психологическая, а не реальная. Это вот с самолётами проблема — они реально летать должны. Видимо именно поэтому новых моделей даже в эпоху «прорыва» что то не видать. В основном модификация того, что успели сделать коммунисты продвигаясь по своему «особому пути» до тех пор, пока не надорвались под тяжестью военных расходов.

Amomum 9 февраля 2019 в 18:04 0

> Задачи настолько специфические что не используют обмена по UART?

Так UART-то работает. И RS-232 или 422 тоже спокойно работал бы. Я полагаю, что оригинальному заказчику просто не нужен был 485.
И с остальными косяками наверняка такая же история.

> Что касается миландра, то зачем им париться. Военные и ФСБшники всё равно купят, у них нет выбора, а ни один коммерческий проект в здравом уме и памяти всё равно их замечательные изделия использовать не будет по причине полной неконкурентоспособности.

> Но мы ушли в сторону. TI к примеру вроде тоже никто не платит, за исходники, примеры и даже даташиты. Но они их делают, потому что их цель сделать полноценный продукт для конкуренции на глобальном рынке.

Насколько мне известно, американские производители микроэлектроники (те же TI в том числе) лет 30-40 точно так же жили за счет ВПК и госзаказа, смогли развернуть производство, подготовить кадры и т.д.
Просто это началось в 60-е. И сейчас у них совсем иная ситуация на рынке, потому что ВПК уже весь поделен и приходится бороться за потребительскую электронику.

У нас же аналогичные процессы только начинаются. И конкуренция у Миландра тоже есть — НИИЭТ, НИИСИ, Модуль (хотя с их продукцией я дела не имел).

Если что, я за Миландр не топлю, не подумайте :) Просто осознаю, что в текущих обстоятельствах действительно сложно сделать лучше. Видно, что они стараются.

> ПО — это полностью их инициатива, за которую им никто напрямую не платит
> — надо избавляться от такого подхода.

Я с вами полностью согласен! Но при этом отмечу, что к моему огромному неудовольствию, очень мало производителей микроэлектроники следует хорошим практикам разработки ПО.
Библиотеки от STM — это просто кладезь маразма, которые прячут другие макросы, которые прячут другие макросы и так без конца, динамическая память льется рекой. Публичного багтрекера нет, о косяках приходится писать на ооооочень медленном форуме где его может быть заметит модератор. Может быть. И перепостит во внутренний багтрекер.

Пока что радует только ARM, которые CMSIS на гитхаб выложили и следят за ним. И Кейл перетаскивают с самодельного компилятора на Clang.

Но в целом эмбеду еще очень далеко до хороших софтовых практик. Юнит-тесты? Пфф, компилируется — значит работает! Код-ревью? А зачем? Вжух-вжух и в продакшен!

progchip666 11 февраля 2019 в 12:38 0

У моего поста не было цели наезда на «Миландр». Скорее даже наоборот, я пытался объяснить что экономико-политическое устройство определяет политику компаний. И другими они в сегодняшних российских реалиях быть и не могут. Их продукция конкурентоспособна только в узком кругу. В условиях самоизоляции нет воможности выхода на глобальные рынки, без этого не опустишь цену. Нет у них ресурса и стимула на «допил» продукта. Немногочисленные клиенты уже изучили их косяки и научились подставлять под них костыли. Новых всё равно нет. В таких условиях лучше направить ресурс на клонирование следующего, более современного камня, чем доводить до ума старый.

> Библиотеки от STM — это просто кладезь маразма

Тут вынужден согласиться. Кроме того замечу что и у ведущих производителей с новыми камнями немало косяков случается, но они их фиксят, выпускают erratы и в следующей ревизии исправляют ошибки

> Насколько мне известно, американские производители микроэлектроники (те же TI в том числе) лет 30-40 точно так же жили за счет ВПК и госзаказа, смогли развернуть производство, подготовить кадры и т.д.

Проблема в том, что мир за 40 лет стал другим. Он стал глобальным и такая тактика уже лет 20 как не работает. Сейчас высокие технологии рождаются в гражданских фирмах, а потом их заимствуют военные. В сегодняшнем мире попытки развития технологий по шаблону 40 летней давности не сработают, а только увеличат отставание и разорит страну.

An_private 8 февраля 2019 в 13:27 +2

> Если вы можете себе позволить перемычку (в идеале – управляемую) между ногами RX и TX, то всё тоже хорошо.

Так она у вас УЖЕ ЕСТЬ. Вам же достаточно повесить nRE на землю и всё — у вас на выходе ресивера появится всё, что вы передаёте.

Amomum 8 февраля 2019 в 13:39 +1

В том конкретном случае мы так сделать не могли, потому что плат уже изготовлено много, они под лаком, а ноги nRE и DE объединены.

Amomum 8 февраля 2019 в 13:45 +1

Но вообще, я не электронщик и не очень понимаю, что будет, если так сделать. Не уверен, что работа этих ног как-то стандартизирована среди разных микросхем.

An_private 8 февраля 2019 в 13:51 +1

> не очень понимаю, что будет, если так сделать

Просто ресивер не выключается и продолжает принимать все данные с шины — как свои, так и чужие.

> Не уверен, что работа этих ног как-то стандартизирована среди разных микросхем.

По сути стандартизирована — практически все драйвера RS485 pin-to-pin compatible.

Amomum 8 февраля 2019 в 14:02 +1

Мысль интересная, надо будет попробовать. С другой стороны, не хочется все время слышать эхо, а если RE и DE управлять отдельно, то придется еще одну ногу потратить.

An_private 8 февраля 2019 в 14:48 +1

> не хочется все время слышать эхо

Почему нет? В прерывании по приходу байта проверяем — мы в передаче или нет. Если нет — обрабатываем. Если в передаче — смотрим — последний это байт или нет, если нет — сразу выходим. При таких скоростях нагрузка на процесор никакая.

> Как бы то ни было, мы не можем опустить RE-DE сразу по входу в это прерывание, потому что так мы «отрежем» от нашего последнего байта стоп-бит или часть стоп-бита.

Вообще-то можем, если на линии RS485 предусмотрели смещение.

Amomum 8 февраля 2019 в 15:01 +1

> Почему нет? В прерывании по приходу байта проверяем — мы в передаче или нет. Если нет — обрабатываем. Если в передаче — смотрим — последний это байт или нет, если нет — сразу выходим. При таких скоростях нагрузка на процесор никакая.

Ну, это понятно, просто на каждый отправленный байт будет по два прерывания.

> Вообще-то можем, если на линии RS485 предусмотрели смещение.

Вас не затруднит раскрыть эту мысль? Боюсь, я не понял, как это поможет.

An_private 8 февраля 2019 в 15:11 +3

Дело в том, что смещение принудительно переводит свободно висящую линию в состояние «1», а стоп-бит — это как раз «1» на линии. Поэтому даже если мы отключаем передатчик от линии во время стоп-бита, то на линии продолжает висеть «1» за счёт смещения.
Главное — чтобы другая сторона не начала передачу во время стоп-бита, иначе будет ошибка фрейминга (не знаю — детектируется она здесь или нет). Но это уже проблемы другой стороны :)

Amomum 8 февраля 2019 в 15:17 +2

Ого. Интересная инфа, спасибо :)
Не уверен, что делать так — это хорошая идея, но инфа интересная.

An_private 8 февраля 2019 в 15:18 +1

> просто на каждый отправленный байт будет по два прерывания

Можем тупо отключить прерывание на приём до тех пор, пока оно не будет нужно.

Amomum 8 февраля 2019 в 15:21 +2

Резонно.

Naves 8 февраля 2019 в 14:54 0

Когда-то программировал для контроллера СКУД, порты на контроллере могли переключаться в режимы 232/422/485. Ридер карточек подключался по двум проводам, вроде как в режиме 485. Если на контроллере включать режим 485, то связи не было. Пришлось на контроллере просто соединить RX-TX по линиям +- в режиме 422. И после отправки команды условно в 20 байт, приходилось первые 20 пришедших байт ответа просто отбрасывать.

Andriyevski 8 февраля 2019 в 16:22 0

Советский союз всегда умел скопировать устройство правильно, дешево и качественно, сейчас же даже скопировать толковые вещи не могут! Видно что создаются вещи только что б выкачивать деньги через дырку в гос. бюджете называемую «Оборонпром», являющиеся приоритетной при составления затрат на год!
Пруф
Я хоть и с Украины, но я б лучше купил русский аналог добротный, чем китайский оригинал, но увы не в этом столетии с такими то чиновниками и программами!

Amomum 8 февраля 2019 в 16:28 +4

Строго говоря, у большинства Миландров честно купленное ядро Cortex — и оно работает как и должно. Почти все проблемы в периферии, а она вроде как не копированная ни с кого, а собственной разработки.

iliasam 9 февраля 2019 в 01:11 +2

Насколько я понял из документации на контроллеры от Миландра, модуль UART там тоже покупной — ARM PL011. Вот его документация: ссылка.
Там прямо структурные схемы в документации от Миландра и ARM практически совпадают.
Я бегло просмотрел документацию на PL011, и не увидел информации о наличии нужного прерывания.

Amomum 9 февраля 2019 в 02:13 +2

А вот это интересная информация! Действительно, выглядит похоже. А где вы в документации Миландра это разглядели?

То есть Миландр даже и не виноват, получается, это ARM — редиски! Вот это поворот.

hhba 9 февраля 2019 в 09:26 0

Да не то чтобы редиски… Повторюсь лишний раз про старый добрый 16550, который в свое время считался «золотым стандартом», и были даже такие странные люди, которые вопили «в новом микроконтроллере от %company name% контроллер UART полностью 16550-совместимый!», так что я не удивлюсь, если это задумка такая — не делать ничего, чем не баловали во времена старых бородатых эмбеддеров.

iliasam 9 февраля 2019 в 11:49 0

Случайно обнаружил документацию на PL011, когда гуглил что-то связанное с UART. Названия регистров там везде одинаковые.
Вот здесь прямо PL011 упоминается, кое-где гуглинг выдает упоминания PL010.
Замечу, что там вроде бы не только UART покупной, но и еще какие-то модули, когда-то на форуме Миландра находил такую информацию.

Amomum 9 февраля 2019 в 17:33 0

Спасибо.

sim2q 10 февраля 2019 в 23:27 0

Очень похоже на работу SPI в stm32f030, там тоже SPI_SR_BSY ловить, но [вроде] можно и в прерывание по приёму слова с того-же пина. По даташиту так получается, но пока не пробовал.

Amomum 11 февраля 2019 в 03:08 0

Как ни странно, ни разу не испытывал потребности в прерываниях от SPI, всегда хватало просто задрать скорость и тупо флаги поллить блокирующе.

Обычно нужно перекинуть два-три байта, вполне можно себе позволить.

sim2q 11 февраля 2019 в 11:10 0

Да у меня тоже не было ещё что бы больше 16 бит за раз, но теперь хочется красиво :)

Amomum 11 февраля 2019 в 13:04 0

А смысл-то в этом есть? :) Вы прикиньте сначала, может мотаться в прерывание и обратно будет даже дольше. Вот если DMA поднять — возможно смысла будет больше.

sim2q 11 февраля 2019 в 14:43 0

Смысл сделать хорошо и забыть:) К примеру сейчас у меня по SPI управляется аналоговый коммутатор для ADC и нужно точно знать, что уже встал нужный канал что бы начать измерение. Но изначально было бы правильно взять более широкий корпус, но уже исторически так:)

hardegor 8 февраля 2019 в 21:24 +1

> и режим FIFO, на мой взгляд, совершенно бесполезный. Если кто-нибудь понимает, зачем он нужен, расскажите, пожалуйста!

Как это бесполезный? Если допустим у вас запрос-ответ протокола 16 или меньше байт — вы закидываете весь пакет в FIFO и вызываете только одно прерывание, когда сработал TXFE и остался последний байт в передатчике. Или например надо передать большой блок данных. У вас получается в 15 раз меньше вызовов прерывания.

> — Оператива делится на два куска, при этом код можно запускать только из второго, DMA может работать только со вторым и при прошивании можно использовать только второй кусок.

Так делается у всех Cortex с Ethernet, эта память работает на частоте ядра и именно туда удобнее складывать данные с Ethernet.

Amomum 8 февраля 2019 в 22:26 +1

> Так делается у всех Cortex с Ethernet, эта память работает на частоте ядра и именно туда удобнее складывать данные с Ethernet.

Подозреваю, что у других Cortex это более доходчиво написано в даташите :)

> Как это бесполезный? Если допустим у вас запрос-ответ протокола 16 или меньше байт — вы закидываете весь пакет в FIFO и вызываете только одно прерывание, когда сработал TXFE и остался последний байт в передатчике. Или например надо передать большой блок данных. У вас получается в 15 раз меньше вызовов прерывания.

Почему-то я был уверен, что FIFO на прием работает как-то не так, но за давностью лет в упор не помню, в чем там была проблема.

Misaka10032 11 февраля 2019 в 11:48 0

> Так делается у всех Cortex с Ethernet, эта память работает на частоте ядра и именно туда удобнее складывать данные с Ethernet.

У STM32F107(Cortex-M3, есть интегрированный MAC) не припомню такого.

hhba 8 февраля 2019 в 23:32 +2

Во-первых спасибо за статью, надо Хабр разбавлять подобным!

Во-вторых все-таки должен сказать, что повторная попытка пнуть лежачего все так же неубедительна. Я чесгря не в курсе был прошлый раз, что вы имеете в виду под «шлейфовым режимом» (т.к. «Миландр» — не моя тема), а оказалось, что это более чем няшка, решающая вопрос весьма элегантно. Но и без него норм, вам @An_private расписал ровно то, о чем я в комментариях в прошлый раз говорил. Наверное мне просто повезло, но в основном всегда можно было сделать петлю уже на «физике», управляя сигналами DE/RO.

В одной из плат была реализована хитрость, деталей которой я уже не помню, когда DE управлялся де-факто потоком данных, правда

для этого использовалась некоторая логика (там как раз пол-CPLD без дела простаивало, туда и пристроили), ну и конечно смещение на стороне «физики» было. Правда это уж совсем костыль, как по мне, к тому же «железячный» (что как бы не круто).

> режим FIFO, на мой взгляд, совершенно бесполезный. Если кто-нибудь понимает, зачем он нужен, расскажите, пожалуйста!

В этом месте я подумал, что вы издеваетесь… Как это зачем?!

Если проблемы были именно с приемом, то там могли быть некорректно работающие или, что вероятнее, некорректно использованные interrupt reasons. У тех же 16550 были весьма кучерявые режимы выдачи прерываний, как-то плохо уже их помню, но хитрости были, да. И это общее место у многих других контроллеров UART. К слову, я лично обнаруживал, что у нативного SerialPort в дотнете threshold входного буфера на некоторых компах работает некорректно, что со слов одного чувака якобы имеет восходит корнями к кривым COM-портам на материнках и не менее кривым драйверам оных.

Amomum 8 февраля 2019 в 23:46 +1

Спасибо :)

Если я правильно помню, то FIFO не зашел, потому что у какого-то девайса был ответ неизвестной заранее длины. Или прерывание не вылезало, когда должно было. Не, не помню, хоть тресни; впечатление осталось, а причину забыл.

Ну и аппаратная петля, конечно, вполне себе вариант, но у нас на момент обнаружения этого косяка было около 50 плат под лаком, с оформленным РКД и военной приемкой :)

hhba 8 февраля 2019 в 23:55 +1

> Ну и аппаратная петля, конечно, вполне себе вариант, но у нас на момент обнаружения этого косяка было около 50 плат под лаком, с оформленным РКД и военной приемкой

Тогда в дело идет упомянутый «шлейфовый режим» — и вперед!

Я только вот чего не пойму — что это за такие лояльные воены, которые на 485-й согласились вместо чего-нибудь фимозного/легаси? Они ж любят всякое эдакое…

Amomum 9 февраля 2019 в 00:01 +1

> Тогда в дело идет упомянутый «шлейфовый режим» — и вперед!

Ну он и пошел, собственно :)

> Я только вот чего не пойму — что это за такие лояльные воены, которые на 485-й согласились вместо чего-нибудь фимозного/легаси? Они ж любят всякое эдакое…

По 485 подключались покупные датчики, которые мы сами выбирали :)

hhba 9 февраля 2019 в 09:22 0

> По 485 подключались покупные датчики, которые мы сами выбирали

Ничоси, и им типа было пофигу, какие там интерфейсы? Вы случаем не на американских военных работали?))))

Amomum 9 февраля 2019 в 18:09 0

> Ничоси, и им типа было пофигу, какие там интерфейсы?

Не могу точно вам ответить. Полагаю, что все было согласовано.

bugdesigner 9 февраля 2019 в 07:32 — 0

Я правильно понял, что "военная приёмка" — это чистой воды профанация? Иначе как объяснить наличие 50 неработающих плат, которые прошли такую приёмку?

hhba 9 февраля 2019 в 09:23 — +1

А почему неработающих? Как раз из текста выше следует, что работающих.

Amomum 9 февраля 2019 в 17:24 — 0

Собсно, мы обнаружили проблему и решили ее (в тот момент — с помощью таймера) существенно раньше, чем собственно приемка должна была быть.

GolikovAndrey 9 февраля 2019 в 11:48 — +1

Видимо под лаком и военной приемкой скрыт еще один баг. В такой организации шины стоп бит и идле шины не отличимы. И выключение драйвера не должно было что то менять. Видимо на шине нет подтяжек, чтобы пустая она была в 1. И эти треть бита нужны были чтобы успели включиться драйверы внешних датчиков.
И вам везет что по приему вы успеваете включить свои драйверы до отключения внешних и шина не проваливается в непонятное ничто.
И да, потенциально вы порождаете конфликты на шине.

Amomum 9 февраля 2019 в 17:27 — 0

Я, наверное, лучше ничего пока говорить не буду, ибо электроника — это уже не моя область, а выходные коллег на этот счет дергать не хочется.

San66 10 февраля 2019 в 10:18 — 0

А может проще было на четырёхпроводку перейти? Линии приёма и передачи будут разделены, или там затейливая топология?

Amomum 10 февраля 2019 в 11:24 — 0

Ну, если можно на 422 перейти, то проще, конечно. Но если покупное изделие хочет 485, то тут ничего не поделать. Особенно, если оно уже 100 раз утверждено и закуплено.

САМОЕ ЧИТАЕМОЕ

Сутки | Неделя | Месяц

Как жить и работать дальше, если вам поставили психиатрический диагноз

+136 49,6k 98 119

Прощай, Google! 15 Альтернативных поисковиков, которые не шпионят, а сажают деревья и раздают воду

+38 38,8k 309 89

Как дочка Роснано, продавшая с Ростехом тысячи камер в школы, делает «российские» камеры с дырявой китайской прошивкой

+279 65,9k 163 284

Обсуждение системы голосования, разработанной ДИТ Москвы

+30 11,3k 20 57

Agile для банка: кейс о том, как выстроить ИТ-подразделение

Мегапост

Если нашли опечатку в посте, выделите ее и нажмите Ctrl+Enter, чтобы сообщить автору.