**РЕФЕРАТ**

Записка 68 стр., 13 таб., 34 рис., 4 лист., 12 источников, 2 прил.

МИКРОКОНТРОЛЛЕР, МИЛАНДР, SPI, UART, CAN, USB

В ходе работы над данным курсовым проектом были исследованы МК К1986ВЕ92QI, отладочная плата для него и интерфейсы взаимодействия двух отладочных плат. Были написаны тестирующие программы для каждого модуля по-отдельности и всех модулей в ходе одного теста.

Тестирующая программа сдержит тесты:

* модуля SSP;
* модуля UART;
* модуля USB;
* модуля CAN;
* полного тестирования (то есть тест всех модулей с выводом результатов тестирования на каждом этапе на экран).

Материалы по курсовой работе представлены в виде графической части, приложений со схемами и отлаженным программным кодом для микроконтроллера и расчетно-пояснительной записки.

**СПИСОК СОКРАЩЕНИЙ**

МК – микроконтроллер

АЦП – аналого-цифровой преобразователь

ЦАП – цифро-аналоговый преобразователь

ОЗУ – оперативное запоминающее устройство

LCD – liquid crystal display (жидкокристаллический дисплей)

SPI - Serial Peripheral Interface (последовательный периферийный интерфейс)

UART - universal asynchronous receiver/transmitter (Асинхронный последовательный интерфейс)

CAN - Controller Area Network (сеть контроллеров)

USB - Universal Serial Bus (универсальная последовательная шина)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc25549180)

[1 Конструкторская часть 7](#_Toc25549181)

[1.1 Краткое описание архитектуры и характеристик К1986ВЕ92QI 7](#_Toc25549182)

[1.2 Описание интерфейсов и контроллеров интерфейсов 15](#_Toc25549183)

[1.2.1 USB 15](#_Toc25549184)

[1.2.2 CAN 21](#_Toc25549185)

[1.2.3 SSP 31](#_Toc25549186)

[1.2.4 UART 38](#_Toc25549187)

[1.3 Описание отладочной платы 43](#_Toc25549188)

[1.3.1 LCD 45](#_Toc25549189)

[1.3.2 Приемопередатчик CAN 46](#_Toc25549190)

[1.3.3 Приемопередатчик RS-232 48](#_Toc25549191)

[1.4 Подключение отладочных плат 50](#_Toc25549192)

[1.5 Алгоритмы работы программы 50](#_Toc25549193)

[1.5.1 Алгоритм работы основной программы 50](#_Toc25549194)

[1.5.2 Алгоритмы инициализации 52](#_Toc25549195)

[1.5.3 Алгоритмы работы прерываний 54](#_Toc25549196)

[1.5.4 Алгоритмы работы тестирующей программы CAN 57](#_Toc25549197)

[1.5.5 Алгоритмы работы тестирующей программы SSP 57](#_Toc25549198)

[1.5.6 Алгоритмы работы тестирующей программы USB 58](#_Toc25549199)

[1.5.7 Алгоритмы работы тестирующей программы UART 59](#_Toc25549200)

[1.5.8 Алгоритмы работы полной тестирующей программы 59](#_Toc25549201)

[2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 64](#_Toc25549202)

[2.1 Разработка программы 64](#_Toc25549203)

[2.2 Программирование flash-памяти микроконтроллера, отладка программы 64](#_Toc25549204)

[2.3 Программирование USB драйвера 65](#_Toc25549205)

[ЗАКЛЮЧЕНИЕ 67](#_Toc25549206)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 68](#_Toc25549207)

ПРИЛОЖЕНИЕ А – Спецификация

ПРИЛОЖЕНИЕ Б – Исходный текст программы

# ВВЕДЕНИЕ

С развитием технологий разработки микросхем, сложные вычислительные задачи могут быть решены с помощью микроконтроллеров, имеющих очень компактные размеры.

АО "ПКК Миландр" — ведущий российский разработчик и производитель изделий микроэлектроники (микроконтроллеры, микропроцессоры, микросхемы памяти, микросхемы приемопередатчиков, микросхемы преобразователей напряжения, радиочастотные схемы), универсальных электронных модулей и приборов промышленного и коммерческого назначения, разработки ПО для современных информационных систем и изделий микроэлектроники. Устройства данной компании по праву пользуются достаточно высокой популярностью на российском и зарубежном рынке.

Цель работы – исследование интерфейсов взаимодействия двух отладочных плат для микросхемы К1986ВЕ92QI (ТСКЯ.469575.002-01 вер. 4) и разработка тестирующих программ для модулей SSP, UART, CAN и USB. Разработка программ взаимодействия двух микроконтроллеров, демонстрирующих работу перечисленных интерфейсов, включая средства для отображения приема/передачи данных по интерфейсам.

# 1 Конструкторская часть

## 1.1 Краткое описание архитектуры и характеристик К1986ВЕ92QI

На основе информации с официального сайта АО "ПКК Миландр" со страницы данного МК может быть получена информация о его основных технических характеристиках. Данная информация является общей для всех микроконтроллеров серии 1986ВЕ9х.

1. Ядро:

* ARM 32-битное RISC-ядро Cortex™-M3 ревизии 2.0, тактовая частота до 80 МГц,
* производительность 1.25 DMIPS/МГц (Dhrystone 2.1) при нулевой задержке памяти;
* блок аппаратной защиты памяти MPU;
* умножение за один цикл, аппаратная реализация деления.

1. Память:

* встроенная энергонезависимая Flash-память программ размером 128 Кбайт;
* встроенное ОЗУ размером 32 Кбайт;
* контроллер внешней шины с поддержкой микросхем памяти СОЗУ, ПЗУ, NAND Flash.

1. Питание и тактовая частота:

* внешнее питание 2,2 ÷ 3,6 В;
* встроенный регулируемый стабилизатор напряжения на 1,8 В для питания ядра;
* встроенные схемы контроля питания;
* встроенный домен с батарейным питанием;
* встроенные подстраиваемые RC генераторы 8 МГц и 40 кГц;
* внешние кварцевые резонаторы на 2 ÷ 16 МГц и 32 кГц;
* встроенный умножитель тактовой частоты PLL для ядра;
* встроенный умножитель тактовой частоты PLL для USB.

1. Режим пониженного энергопотребления:

* режимы Sleep, Deep Sleep и Standby;
* батарейный домен с часами реального времени и регистрами аварийного сохранения.

1. Аналоговые модули:

* два 12-разрядных АЦП (до 16 каналов);
* температурный датчик;
* двухканальный 12-разрядный ЦАП;
* встроенный компаратор.

1. Периферия:

* контроллер DMA с функциями передачи Периферия-Память, Память-Память;
* два контроллера CAN интерфейса;
* контроллер USB интерфейса с функциями работы Device и Host;
* контроллеры интерфейсов UART, SPI, I2C;
* три 16-разрядных таймер-счетчика с функциями ШИМ и регистрации событий;
* до 96 пользовательских линий ввода-вывода.

1. Отладочные интерфейсы:

* последовательные интерфейсы SWD и JTAG.

1. Тип корпуса - LQFP64.
2. Ближайший аналог - STM32F103x.
3. Температурный диапазон – минус 45 °С ...+85°С

Таким образом, микроконтроллер может быть использован для решения широкого спектра задач, так как обладает внушительными характеристиками.

На рисунке 1 представлена схема расположения выводов данного микроконтроллера. Стоит заметить, что почти все выводы с портов ввода/вывода имеют альтернативные функции, не представленные на данном рисунке, и могут быть переопределены для выполнения функций различных модулей

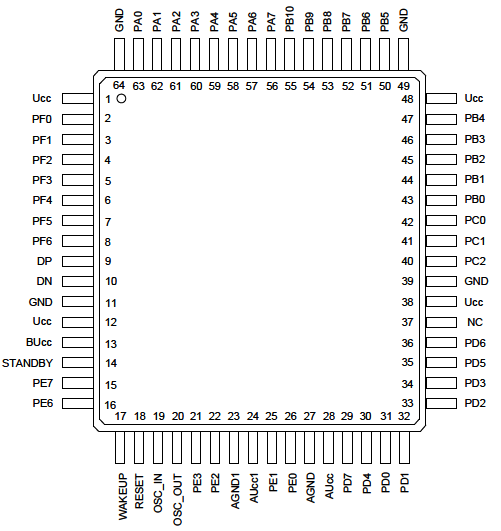


Рисунок 1 – Схема выводов микроконтроллера

Назначение линий портов микроконтроллера приведено в таблице 1. Для того, чтобы линии порта перешли под управление того или иного периферийного блока, необходимо задать для выбранных линий выполняемую функцию и настройки. Обратим внимание на то, что микроконтроллер имеет только 64 вывода, поэтому не каждая линия порта может быть связана с выводом микросхемы.

Таблица 1 - Функции линий портов микроконтроллера К1986ВЕ92QI

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Линия** | **Вывод** |  | **Цифровая функция** |  | **Аналоговая функция** |
| **Основная** | **Альтернат.** | **Переопред.** |
| **1** | **2** | **3** | **4** | **5** | **6** |
|  | | | Порт A |  | |
| PA0 | 63 | DATA0 | EXT\_INT1 | – | – |
| PA1 | 62 | DATA1 | TMR1\_CH1 | TMR2\_CH1 | – |
| PA2 | 61 | DATA2 | TMR1\_CH1N | TMR2\_CH1N | – |
| PA3 | 60 | DATA3 | TMR1\_CH2 | TMR2\_CH2 | – |
| PA4 | 59 | DATA4 | TMR1\_CH2N | TMR2\_CH2N | – |
| PA5 | 58 | DATA5 | TMR1\_CH3 | TMR2\_CH3 | – |
| PA6 | 57 | DATA6 | CAN1\_TX | UART1\_RXD | – |
| PA7 | 56 | DATA7 | CAN1\_RX | UART1\_TXD | – |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | Порт B |  | |
| PB0 | 43 | DATA16 | TMR3\_CH1 | UART1\_TXD | – |
| PB1 | 44 | DATA17 | TMR3\_CH1N | UART2\_RXD | – |
| PB2 | 45 | DATA18 | TMR3\_CH2 | CAN1\_TX | – |
| PB3 | 46 | DATA19 | TMR3\_CH2N | CAN1\_RX | – |
| PB4 | 47 | DATA20 | TMR3\_BLK | TMR3\_ETR | – |
| PB5 | 50 | DATA21 | UART1\_TXD | TMR3\_CH3 | – |
| PB6 | 51 | DATA22 | UART1\_RXD | MR3\_CH3N | – |
| PB7 | 52 | DATA23 | nSIROUT1 | TMR3\_CH4 | – |
| PB8 | 53 | DATA24 | COMP\_OUT | TMR3\_CH4N | – |
| PB9 | 54 | DATA25 | nSIRIN1 | EXT\_INT4 | – |
| PB10 | 55 | DATA26 | EXT\_INT2 | nSIROUT1 | – |
|  |  |  | Порт C |  | |
| PC0 | 42 | – | SCL1 | SSP2\_FSS | – |
| PC1 | 41 | OE | SDA1 | SSP2\_CLK | – |
| PC2 | 40 | WE | TMR3\_CH1 | SSP2\_RXD | – |
|  |  |  | Порт D |  | |
| PD0 | 31 | TMR1\_CH1N | UART2\_RXD | TMR3\_CH1 | ADC0\_REF+ |
| PD1 | 32 | TMR1\_CH1 | UART2\_TXD | TMR3\_CH1N | ADC1\_REF- |
| PD2 | 33 | BUSY1 | SSP2\_RXD | TMR3\_CH2 | ADC2 |
| PD3 | 34 | – | SSP2\_FSS | TMR3\_CH2N | ADC3 |

Окончание таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** |
| PD4 | 30 | TMR1\_ETR | nSIROUT2 | TMR3\_BLK | ADC4 |
| PD5 | 35 | CLE | SSP2\_CLK | TMR2\_ETR | ADC5 |
| PD6 | 36 | ALE | SSP2\_TXD | TMR2\_BLK | ADC6 |
| PD7 | 29 | TMR1\_BLK | nSIRIN2 | UART1\_RXD | ADC7 |
|  |  |  | Порт E |  | |
| PE0 | 26 | ADDR16 | TMR2\_CH1 | CAN1\_RX | DAC2\_OUT |
| PE1 | 25 | ADDR17 | TMR2\_ CH1N | CAN1\_TX | DAC2\_REF |
| PE2 | 22 | ADDR18 | TMR2\_CH3 | TMR3\_CH1 | COMP\_IN1 |
| PE3 | 21 | ADDR19 | TMR2\_CH3N | TMR3\_CH1N | COMP\_IN2 |
| PE6 | 16 | ADDR22 | CAN2\_RX | TMR3\_CH3 | OSC\_IN32 |
| PE7 | 15 | ADDR23 | CAN2\_TX | TMR3\_CH3N | OSC\_OUT32 |
|  |  |  | Порт F |  | |
| PF0 | 2 | ADDR0 | SSP1\_TXD | UART2\_RXD | – |
| PF1 | 3 | ADDR1 | SSP1\_CLK | UART2\_TXD | – |
| PF2 | 4 | ADDR2 | SSP1\_FSS | CAN2\_RX | – |
| PF3 | 5 | ADDR3 | SSP1\_RXD | CAN2\_TX | – |
| PF4 | 6 | ADDR4 | – | – | – |
| PF5 | 7 | ADDR5 | – | – | – |
| PF6 | 8 | ADDR6 | TMR1\_CH1 | – | – |

На рисунке 2 изображена структурная блок-схема микроконтроллера К1986ВЕ92QI, наглядно представляющая наличествующие периферийные устройства и их взаимодействие. Используемые обозначения представлены в таблице 2.

Таблица 2 – таблица обозначений функциональных блоков К1986ВЕ92QI

|  |  |
| --- | --- |
| **Блок** | **Описание** |
| Cortex-M3 RISC CORE | Процессорное ядро ARM Cortex-M3 архитектуры RISC |
| DMA | Контроллер прямого доступа в память |
| Interrupt | Контроллер прерываний |
| System timer | Системный таймер |
| JTAG/SW debug | Отладочный модуль через интерфейс JTAG/SW |
| AMBA AHB Bus Matrix | Шинная матрица для связи высокоскоростных внутренних компонентов |
| AHB APB Bridge | Мост для связи с периферией |
| Flash | Модуль памяти Flash |

Окончание таблицы 2

|  |  |
| --- | --- |
| RAM | Модуль памяти RAM |
| ROM | Модуль памяти ROM |
| External System Bus | Внешняя системная шина |
| System Clock Manager | Модуль системного тактирования |
| UART | Контроллер UART |
| SPI | Контроллер SPI |
| BKP Controller | Контроллер резервных данных |
| Power Detector | Модуль управления питанием |
| I2C | Контроллер I2C |
| GPIO | Интерфейс ввода/вывода общего назначения |
| 16 Timer | 16-разрядный таймер |
| ADC Controller | Контроллер аналого-цифрового преобразователя |
| CAN | Контроллер CAN |
| DAC Controller | Контроллер цифро-аналогового преобразователя |
| USB | Контроллер USB |
| Controller Comparator | Контроллер компаратора |
| WDT Controller | Контроллер сторожевого таймера |
| CPU PLL | Фазовая автоподстройка частоты для процессорного ядра |
| USB PLL | Фазовая автоподстройка частоты для USB |
| HSI | Высокоскоростной внутренний генератор тактовой частоты |
| LSI | Низкоскоростной внутренний генератор тактовой частоты |
| HSE | Высокоскоростной внешний генератор тактовой частоты |
| LSE | Низкоскоростной внешний генератор тактовой частоты |
| Real Time Clock BKP memory | Резервная память |
| LDO Cap Less | Регулятор напряжения |
| ADC | АЦП |
| DAC | ЦАП |
| PSY USB | Дескриптор USB |
| Comparator | Компаратор |
| IWDT | Независимый сторожевой таймер |

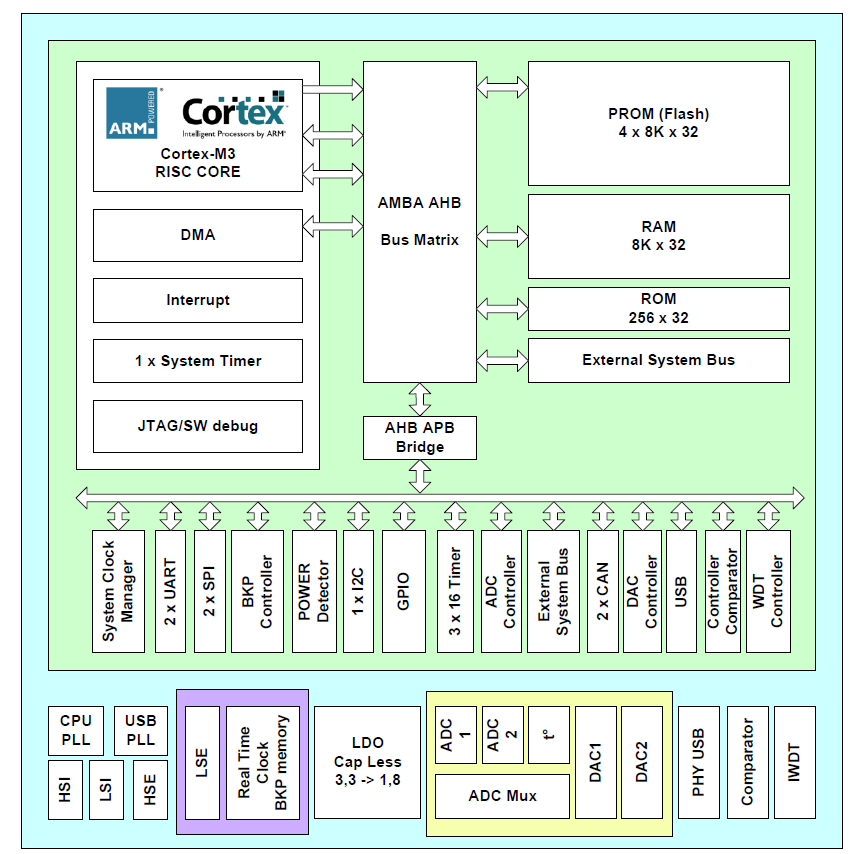


Рисунок 2 – Структурная блок-схема микроконтроллера К1986ВЕ92QI

Процессорное ядро имеет три системных шины:

* I Code – шина выборки инструкций;
* D Code – шина выборки данных, расположенных в коде программы;
* S Bus – шина выборки данных, расположенных в области ОЗУ.

Также в микроконтроллере реализован контроллер прямого доступа в память (DMA), который осуществляет выборку через шину DMA Bus. Структурная схема организации памяти представлена на рисунке 3.

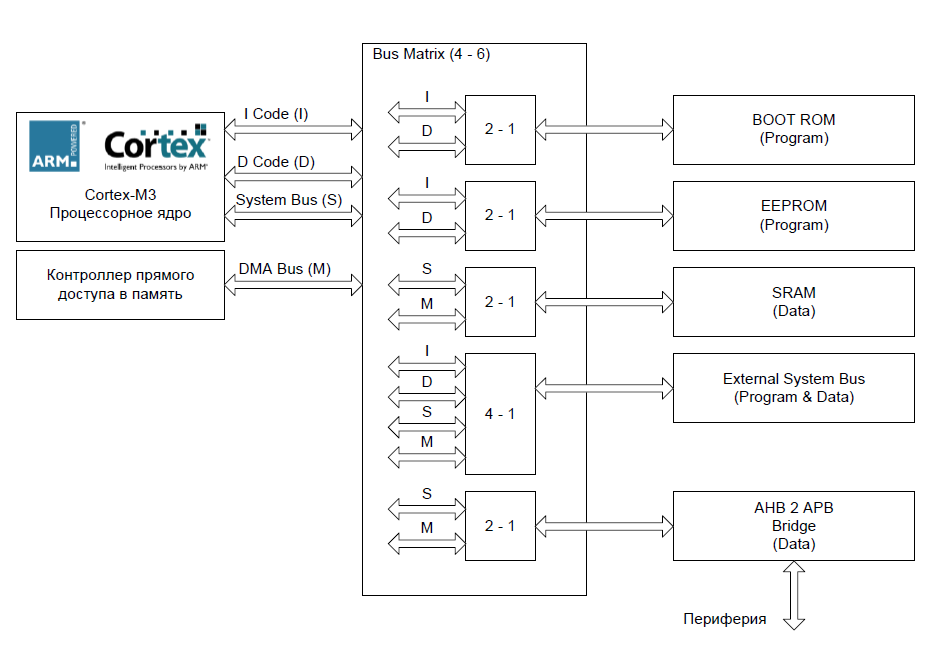


Рисунок 3 – Структурная схема организации памяти

Все адресное пространство микроконтроллера едино и имеет максимальный объем 4 Гбайт. В данное адресное пространство отображаются различные модули памяти и периферии.

По умолчанию для записи программ используется область памяти 0x08000000 - 0x0801FFFFF внутренней Flash памяти.

После включения питания и снятия внутренних (POR) и внешних (RESET) сигналов сброса, микроконтроллер начинает выполнять программу из загрузочной области ПЗУ BOOT ROM. В загрузочной программе микроконтроллер определяет, в каком из режимов он будет функционировать, и переходит в этот режим. Режим функционирования определяется внешними выводами MODE[2:0], при этом перед опросом состояния этих выводов, для них включается внутренняя подтяжка к шине «Общий» (встроенные резисторы подтяжки к шине «Общий» имеют сопротивление ~50 кОм). Также устанавливается бит FPOR в регистре BKP\_REG\_0E, который может быть сброшен только при отключении основного питания UCC. После перезапуска микроконтроллера уровни на выводах MODE[2:0] не влияют на режим функционирование микроконтроллера, если установлен бит FPOR.

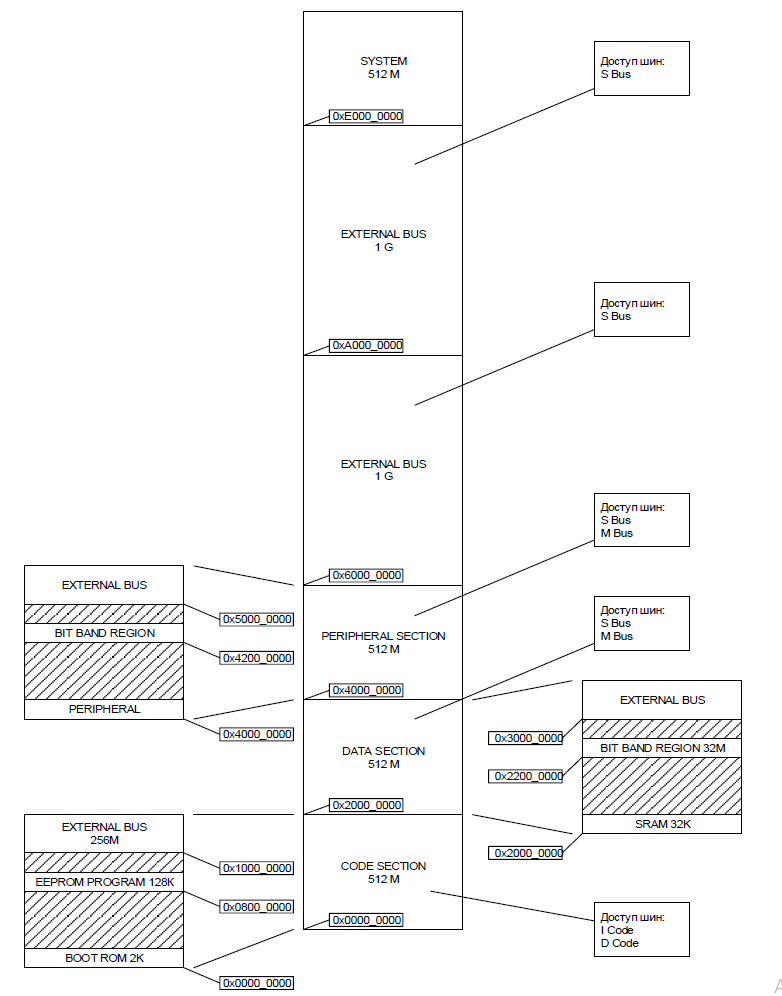


Рисунок 4 - Карта распределения основных областей памяти

При работе с отладочной платой MODE[2:0] устанавливается с помощью переключателей на плате. Режимы запуска МК для JTAG представлены в таблице 3.

Таблица 3 – Режимы первоначального запуска микроконтроллера

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MODE**  **[2:0]** | **Стартовый адрес** | **Описание** | **Порты** | **Описание выводов интерфейса** |
| 000 | 0х0800\_0000 | Процессор начинает выполнять программу из внутренней Flash-памяти программ. При этом установлен отладочный интерфейс JTAG\_B | PD2/JB\_TRST  PD1/JB\_TCK  PD0/JB\_TMS  PD3/JB\_TDI  PD4/JB\_TDO | В качестве выводов интерфейса используются выводы порта D, совмещенные с каналами АЦП, выводами каналов Таймера 1 и 3, UART2 и SSP2, использование которых при отладке запрещено |
| 001 | 0х0800\_0000 | Процессор начинает выполнять программу из внутренней Flash-памяти программ. При этом разрешается работа отладочного интерфейса JTAG\_А | PB4/JA\_TRST  PB2/JA\_TCK  PB1/JA\_TMS  PB3/JA\_TDI  PB0/JA\_TDO | В качестве выводов интерфейса используются выводы порта B, совмещенные с выводами данных внешней системной шины, выводами таймера 3, выводами UART1 и UART2 и CAN1, использование которых при отладке запрещено |

## 1.2 Описание интерфейсов и контроллеров интерфейсов

Особый интерес в данной работе представляют внешние интерфейсы работы, так как именно они будут использоваться для передачи информации между двумя отладочными платами. В данном микроконтроллере представлено 4 интерфейсных модуля, которые могут использоваться для передачи данных между двумя МК:

* Контроллер интерфейса MDR\_USB
* Контроллер интерфейса MDR\_CAN
* Контроллер MDR\_SSP
* Контроллер MDR\_UART

Рассмотрим каждый из модулей более детально с описанием интерфейса в целом.

### 1.2.1 USB

#### 1.2.1.1 Описание интерфейса

USB – последовательный интерфейс, используемый для подключения периферийных устройств. Соответственно, существуют понятие «главное устройство» (хост, он управляет обменом данными через интерфейс, выступает инициатором обмена) и «периферийное устройство» (клиент, в процессе обмена данными «подчиняется» хосту).

Логика работы у хоста и клиента принципиально отличается, соответственно нельзя напрямую соединять устройства «хост – хост» и «клиент – клиент».

Физически интерфейс USB использует 4 провода: «земля (GND)», «+5В (VBUS)», «D+», «D-». Первые два могут использоваться для питания периферийного устройства (максимальный ток 500 мА). Два последних служат для передачи данных.

На логическом уровне, обмен данными происходит через некоторые логические, виртуальные каналы внутри одного физического USB интерфейса. Такие каналы называют «Конечными точками» (EndPoints).

Конечные точки (каналы) бывают 4 видов:

Control – данный тип канала используется хостом для управления периферийным устройством. Хотя иногда данный тип канала используется для передачи данных.

Bulk — данный тип канала используется для обмена данными. Гарантирование целостности данных и гарантированная доставка данных для данного типа канала реализована «в железе». Однако скорость передачи данных по такому каналу ограничена.

Isochronous — данный тип канала в основном используется для обмена потоковыми данными. Целостность и доставка данных не контролируются, зато скорость значительно выше чем для Bulk каналов.

Interrupt – используются для реализации подобия «прерываний». Такие «прерывания» являются логическими, и никак напрямую не связанны с аппаратными прерываниями МК или прерываниями ОС.

Минимальная реализация USB устройства требует наличие всего одного Control канала (так называемая «нулевая конечная точка»). Остальные типы каналов, как и их количество определяет разработчик устройства исходя из функций устройства.

#### 1.2.1.2 Общее описание контроллера интерфейса

Контролер USB реализует функции контроллера функционального устройства (Device) и управляющего устройства (Host) в соответствии со спецификацией USB 2.0.

Контроллер USB поддерживает следующие возможности: режимы работы Full Speed (12 Мбит/с) и Low Speed (1.5 Мбит/с), контроль ошибок с помощью циклического избыточного кода (CRC), NRZI код приема/передачи, управляющая (Control), сплошная (Bulk), изохронная (Isochronous) передачи и передача по прерываниям (Interrupt), конфигурирование USB Device от 1-й до 4-х оконечных точек; каждая оконечная точка USB Device имеет собственную память FIFO размером 64 байта. USB Host поддерживает до 16 оконечных точек. Возможности USB Host: FIFO размером 64 байта; автоматическая отправка SOF пакетов; вычисление оставшегося во фрейме времени.

Контроллер USB может быть сконфигурирован как USB Host или как USB Device. Конфигурация задается битом HOST\_MODE в регистре HSCR (0 – режим Device, 1 – режим Host). Прием/передача через физический интерфейс USB разрешаются установкой бит EN\_RX и EN\_TX в этом же регистре. В режиме приема имеется возможность отключить передатчик в целях экономии потребления (EN\_TX=0). Отключение всего блока в целом осуществляется при EN\_RX=0.

В режиме Device параметры шины задаются в регистре SC. Скорость задается битом SCFSR (0 – 1,5 Мбит/с, 1 – 12 Мбит/с), полярность битом SCFSP (0 – Low speed, 1 – Full speed) этого регистра.

В режиме Host параметры шины задаются в регистре HTXLC. Скорость задается битом FSLR (0 – 1,5 Мбит/с, 1 – 12 Мбит/с), полярность битом FSPL (0 – Low speed, 1 – Full speed) этого регистра.

В режиме Host контроллер автоматически определяет подключение или отключение устройства к шине. Бит CONEV регистра USB\_HSI устанавливается в 1 при возникновении одного из событий.

#### 1.2.1.3 Задание адреса и инициализация оконечных точек

Функциональный адрес устройства USB задается в регистре SA.

Для инициализации конечной точки в первую очередь необходимо установить бит глобального разрешения всех оконечных точек (SCGEN = 1 в регистре SC). Биты EPEN в регистрах SEP[x].CTRL должны быть установлены, чтобы разрешить соответствующую оконечную точку. Если предполагается использовать изохронный тип передачи оконечной точки, то необходимо установить бит EPISOEN в соответствующем регистре SEP[x].CTRL.

#### 1.2.1.4 Описание регистров USB

Описание регистров с базовым адресом, их названием и описанием контроллера USB представлено в таблице 4.

Таблица 4 – Описание регистров контроллера USB

|  |  |  |
| --- | --- | --- |
| **Базовый Адрес** | **Название** | **Описание** |
| 0x4001\_0000 | MDR\_USB | Контроллер USB интерфейса |
| **Смещение** |  |  |
| 0x380 | MDR\_USB->HSCR | Общее управление для контроллера USB интерфейса |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| 0x384 | MDR\_USB->HSVR | Версия аппаратного контроллера USB интерфейса |
|  | **Контроллер HOST** |  |
| 0x00 | MDR\_USB->HTXC | Регистр управления передачей пакетов со стороны хоста |
| 0x04 | MDR\_USB->HTXT | Регистр задания типа передаваемых пакетов со стороны хоста |
| 0x08 | MDR\_USB->HTXLC | Регистр управления линиями шины USB |
| 0x0C | MDR\_USB->HTXSE | Регистр управление автоматической отправки SOF |
| 0x10 | MDR\_USB->HTXA | Регистр задания адреса устройства для отправки пакета |
| 0x14 | MDR\_USB->HTXE | Регистр задания номера оконечной точки для отправки пакета |
| 0x18 0x1C | MDR\_USB->HFN\_L MDR\_USB->HFN\_H | Регистр задания номера фрейма для отправки SOF |
|  |  |  |
| 0x20 | MDR\_USB->HSI | Регистр флагов событий контроллера хост. |
| 0x24 | MDR\_USB->HIM | Регистра флагов разрешения прерываний по событиям контролера хоста |
|  |  |  |
| 0x28 | MDR\_USB->HRXS | Регистр состояния очереди приема данных хоста |
| 0x2C | MDR\_USB->HRXP | Регистр отображения PID принятого пакета |
| 0x30 | MDR\_USB->HRXA | Регистр отображения адреса устройства, от которого принят пакет |
| 0x34 | MDR\_USB->HRXE | Регистр отображения номер оконечной точки, от которой принят пакет |
| 0x38 | MDR\_USB->HRXCS | Регистр отображения состояния подсоединения устройства |
| 0x3C | MDR\_USB->HSTM | Регистр расчета времени фрейма |
| 0x80 | MDR\_USB->HRXFD | Данные очереди приема |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| 0x88 0x8C | MDR\_USB->HRXFDC\_L MDR\_USB->HRXFDC\_H | Число принятых данных в очереди |
| 0x90 | MDR\_USB->HRXFC | Управление очередью приема |
|  |  |  |
| 0xC0 | MDR\_USB->HTXFD | Данные для передачи |
| 0xD0 | MDR\_USB->HTXFC | Управление очередью передачи |
|  |  |  |
|  | **Контроллер SLAVE** |  |
| 0x100  0x110  0x120  0x130 | MDR\_USB->SEP[x].CTRL | Управление очередью нулевой оконечной точки |
| 0x104  0x114  0x124  0x134 | MDR\_USB->SEP[x].STS | Состояние оконечной точки |
| 0x108  0x118  0x128  0x138 | MDR\_USB->SEP[x].TS | Состояние типа передачи оконечной точки |
| 0x10C  0x11C  0x12C  0x13C | MDR\_USB->SEP[x].NTS | Состояние передачи NAK оконечной точки |
| 0x140 | MDR\_USB->SC | Управление контроллеров SLAVE |
| 0x144 | MDR\_USB->SLS | Отображение состояния линий USB шины |
| 0x148 | MDR\_USB->SIS | Флаги событий контроллера SLAVE |
| 0x14C | MDR\_USB->SIM | Флаги разрешения прерываний от контроллера SLAVE |
| 0x150 | MDR\_USB->SA | Функциональный адрес контроллера |

Окончание таблицы 4

|  |  |  |
| --- | --- | --- |
| 0x154 0x158 | MDR\_USB->SFN\_L MDR\_USB->SFN\_H | Номер фрейма |
| 0x180  0x200  0x280  0x300 | MDR\_USB->SEP[x].RXFD | Принятые данные оконечной точки |
| 0x188 0x18C  0x208 0x20C  0x288 0x28C  0x308  0x30C | MDR\_USB->SEP[x].RXFDC\_L MDR\_USB->SEP[x].RXFDC\_H | Число данных в оконечной точке |
| 0x190  0x210  0x290  0x310 | MDR\_USB->SEP[x].RXFC | Управление очередью приема оконечной точки |
| 0x1C0  0x240  0x2C0  0x340 | MDR\_USB->SEP[x].TXFD | Данные для передачи через оконечную точку |
| 0x1D0  0x250  0x2D0  0x350 | MDR\_USB->SEP[x].TXFDC | Управление очередью передачи оконечной точки |

#### 1.2.1.5 Описание библиотечных функций контроллера USB

Для работы с USB Миландром предусмотрена специальная библиотека, которая отвечает за работу модуля в микроконтроллере. В листинге 1 представлены функции данной библиотеки, используемые для работы с контроллером USB в МК в данном курсовом проекте, с комментариями на русском языке. Функции для работы с регистрами представлены в единичном экземпляре, функции для остальных регистров имеют аналогичную структуру.

Листинг 1 – Основные функции библиотеки USB

/\*\*

\* @brief Инициализация предделителя частоты для USB

\* @param USB\_Clock\_InitStruct: указатель на структуру USB\_Clock\_TypeDef

\* @retval None

\*/

void USB\_BRGInit(const USB\_Clock\_TypeDef\* USB\_Clock\_InitStruct)

/\*\*

\* @brief Перезапуск USB

\* @param None

\* @retval None

\*/

void USB\_Reset(void)

/\*\*

\* @brief Возвращает содержимое регистра USB\_HSCR

\* @param None

\* @retval Содержимое USB\_HSCR

\*/

uint32\_t USB\_GetHSCR(void)

/\*\*

\* @brief Устанавливает содержимое регистра USB\_HSCR

\* @param Значение HSCR

\* @retval None

\*/

void USB\_SetHSCR(uint32\_t RegValue)

/\*\*

\* @brief Возвращает версию контроллера USB

\* @param None

\* @retval USB\_Version\_TypeDef структура с информацией о версии контроллера USB

\*/

USB\_Version\_TypeDef USB\_GetHSVR(void)

/\*\*

\* @brief Возвращает значение регистра USB\_SEPx.CTRL

\* @param EndPointNumber: Выбор USB End point устройства

\* @retval Значение регистра USB\_SEPx.CTRL

\*/

uint32\_t USB\_GetSEPxCTRL(USB\_EP\_TypeDef EndPointNumber)

### 1.2.2 CAN

#### 1.2.2.1 Описание интерфейса

CAN (англ. Controller Area Network — сеть контроллеров) — стандарт промышленной сети, ориентированный, прежде всего, на объединение в единую сеть различных исполнительных устройств и датчиков. Режим передачи — последовательный, широковещательный, пакетный. CAN разработан компанией Robert Bosch GmbH в середине 1980-х и в настоящее время широко распространён в промышленной автоматизации, технологиях «умного дома», автомобильной промышленности и многих других областях. Стандарт для автомобильной автоматики.

Информация на шине представлена в виде фиксированных сообщений различной, но ограниченной длины. Когда шина свободна, любой подключенный узел может начать передавать новое сообщение. При передаче информации с помощью протокола CAN используется четыре типа пакетов:

* пакет данных (data frame) — передаёт данные, является самым часто используемым пакетом;
* пакет удаленного запроса (remote frame) — служит для запроса на передачу кадра данных с тем же идентификатором;
* пакет перегрузки (overload frame) — обеспечивает промежуток между кадрами данных или запроса;
* пакет ошибки (error frame) — передаётся узлом, обнаружившим в сети ошибку.

Пакет данных состоит из 7 различных полей:

* "начало пакета" (SOF-start of frame);
* "поле арбитража" (arbitration field);
* "поле контроля" (control field);
* "поле данных" (data field);
* "поле CRC" (CRC field);
* "поле подтверждения" (ACK field);
* "конец пакета" (end of frame).

Поле данных может иметь нулевую длину.

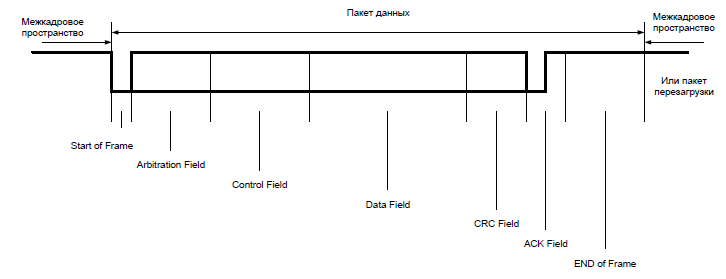


Рисунок 5 – Пакет сообщения CAN

В терминах протокола CAN логическая единица называется рецессивным битом, а логический ноль называется доминантным битом. Во всех случаях доминантный бит будет подавлять рецессивный. То есть, если несколько узлов выставят на шину рецессивный бит, а один – доминантный, то обратно всеми узлами будет считан доминантный бит.

При свободной шине любой узел может начинать передачу в любой момент. В случае одновременной передачи кадров двумя и более узлами проходит арбитраж доступа: передавая идентификатор, узел одновременно проверяет состояние шины. Если при передаче рецессивного бита принимается доминантный — считается, что другой узел передаёт сообщение с большим приоритетом, и передача откладывается до освобождения шины. Таким образом, в отличие, например, от Ethernet в CAN не происходит непроизводительной потери пропускной способности канала при коллизиях. Цена этого решения — возможность того, что сообщения с низким приоритетом никогда не будут переданы.

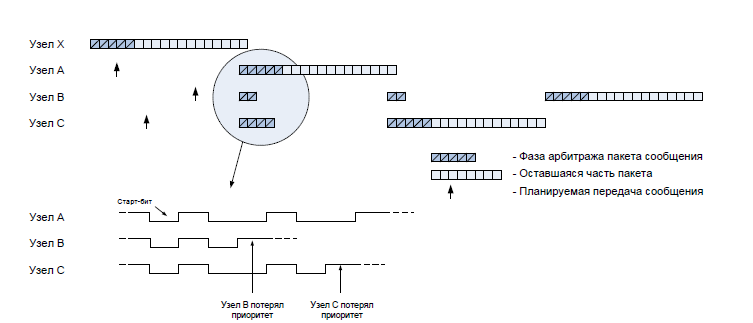


Рисунок 6 – Арбитраж на шине CAN

#### 1.2.2.2 Общее описание контроллера интерфейса

В микроконтроллере реализовано два независимых контроллера интерфейса CAN. Они являются полнофункциональными CAN-узлами, отвечающими требованиям к активным и пассивным устройствам CAN 2.0A и 2.0B и поддерживающими передачу данных на скорости не более 1 Мбит/сек.

Интерфейс CAN позволяет обмениваться сообщениями в сети равноправных устройств. При передаче сообщения в сети CAN все узлы сети получают это сообщение. В сообщении передается уникальный идентификатор и данные. Все сообщения в протоколе CAN могут содержать не более восьми байтов данных. При возникновении коллизий (одновременная передача сообщений различными узлами) при передаче идентификатора происходит арбитраж, и узел передающий сообщение с большим номером идентификатора уступает сеть узлу передающему сообщение с меньшим номером идентификатора.

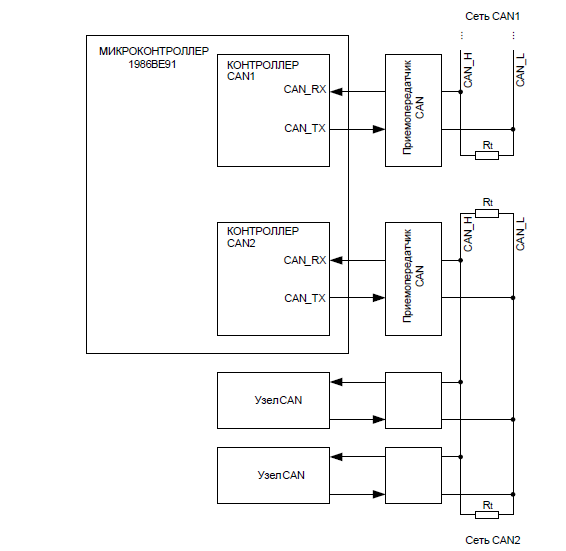


Рисунок 7 - Структурная схема организации сети CAN

Особенности:

* поддержка CAN протокола версии CAN 2.0 A и B;
* скорость передачи до 1 Мбит/с;
* 32 буфера приема/передачи;
* поддержка приоритетов сообщений;
* 32 фильтра приема;
* маскирование прерываний.

#### 1.2.2.3 Прием и передача сообщений

На рисунке 8 представлена структурная блок-схема контроллера CAN

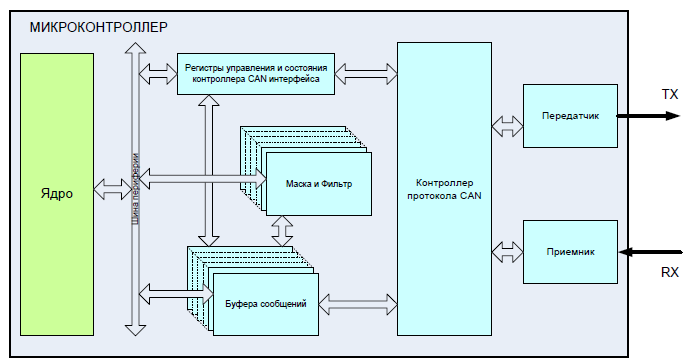


Рисунок 8 - Структурная блок-схема контроллера CAN

Для передачи сообщения необходимо в разрешенный для работы и конфигурируемый на передачу буфер записать сообщение для передачи (задать значения регистрам CAN\_BUF[x].ID, CAN\_BUF[x].DLC, CAN\_BUF[x].DATAL и CAN\_BUF[x].DATAH), после чего установить бит TX\_REQ. После установки этого бита сообщение будет поставлено в очередь на отправку. После отправки сообщения бит TX\_REQ будет автоматически сброшен. Если в нескольких буферах есть сообщения на отправку, то порядок отправки определяется по полю PRIOR\_0. Если у сообщения бит PRIOR\_0 выставлен в ноль, то оно отправляется в первую очередь. Если есть несколько сообщений с одинаковым приоритетом, то порядок отправки определяется порядковым номером буфера, буфер с меньшим порядковым номером имеет больший приоритет. Значение полей ID для выбора порядка отправки в рамках контроллера CAN (одного узла) значения не имеет. По ID выбирается приоритет между различными узлами.

Для приема сообщений необходимо иметь свободные и разрешенные для работы буфера, сконфигурированные на прием сообщений. При этом если по шине CAN будут передаваться сообщения от других узлов, они будут сохраняться в этих буферах.

#### 1.2.2.4 Описание регистров CAN

Описание регистров с базовым адресом, их названием и описанием контроллера CAN представлено в таблице 5.

Таблица 5 – Описание регистров управления контроллера CAN

|  |  |  |
| --- | --- | --- |
| **Базовый адрес** | **Название** | **Описание** |
| 0x4000\_0000 | MDR\_CAN1 | Контроллер интерфейса CAN1 |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| 0x4000\_8000 | MDR\_CAN2 | Контроллер интерфейса CAN2 |
| Смещение |  |  |
| 0x00 | CONTROL | MDR\_CANx->CONTROL  Регистр управление контроллером CAN |
| 0x04 | STATUS | MDR\_CANx->STATUS  Регистр состояния контроллера CAN |
| 0x08 | BITTMNG | MDR\_CANx->BITTMNG  Регистр задания скорости работы |
| 0x10 | INT\_EN | MDR\_CANx->INT\_EN  Регистр разрешения прерываний контроллера |
|  |  |  |
| 0x1C | OVER | MDR\_CANx->OVER  Регистр границы счетчика ошибок |
|  |  |  |
| 0x20 | RXID | MDR\_CANx->RXID  Регистр принятого ID сообщения |
| 0x24 | RXDLC | MDR\_CANx->RXDLC  Регистр принятого DLC сообщения |
| 0x28 | RXDATAL | MDR\_CANx->RXDATAL  Регистр принятых данных |
| 0x2C | RXDATAH | MDR\_CANx->RXDATAH  Регистр принятых данных |
|  |  |  |
| 0x30 | TXID | MDR\_CANx->TXID  Регистр передаваемого ID сообщения |
| 0x34 | TXDLC | MDR\_CANx->TXDLC  Регистр передаваемого DLC сообщения |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| 0x38 | DATAL | MDR\_CANx->TXDATAL  Регистр передаваемых данных |
| 0x3C | DATAH | MDR\_CANx->TXDATAH  Регистр передаваемых данных |
|  |  |  |
| 0x40 | BUF\_CON[0] | MDR\_CANx->BUF\_CON  Регистр управления буфером 01 |
|  | … |  |
| 0xBC | BUF\_CON[31] | MDR\_CANx->BUF\_CON  Регистр управления буфером 32 |
|  |  |  |
| 0xC0 | INT\_RX | Флаги разрешения прерываний от приемных буферов  MDR\_CANx->INT\_RX |
| 0xC4 | RX | Флаги RX\_FULL от приемных буферов MDR\_CANx->RX |
| 0xC8 | INT\_TX | Флаги разрешения прерываний от передающих буферов MDR\_CANx->INT\_TX |
| 0xCC | TX | Флаги ~TX\_REQ от передающих буферов MDR\_CANx->TX |
|  |  |  |
| 0x200 | CAN\_BUF[0].ID | MDR\_CANx->CAN\_BUF[x].ID ID сообщения буфера 01 |
| 0x204 | CAN\_BUF[0].DLC | MDR\_CANx->CAN\_BUF[x].DLC DLC сообщения буфера 01 |
| 0x208 | CAN\_BUF[0].DATAL | MDR\_CANx->CAN\_BUF[x].DATAL  Данные сообщения буфера 01 |
| 0x20C | CAN\_BUF[0].DATAH | MDR\_CANx->CAN\_BUF[x].DATAH  Данные сообщения буфера 01 |

Окончание таблицы 5

|  |  |  |
| --- | --- | --- |
| 0x210 | CAN\_BUF[1].ID | MDR\_CANx->CAN\_BUF[x].ID ID сообщения буфера 02 |
|  | … |  |
| 0x3FC | CAN\_BUF[31].DATAH | MDR\_CANx->CAN\_BUF[x].DATAH  Данные сообщения буфера 32 |
|  |  |  |
| 0x500 | CAN\_BUF\_FILTER[0].MA  SK | MDR\_CANx-  >CAN\_BUF\_FILTER[x].MASK Маска для приема сообщения в буфер 01 |
| 0x504 | CAN\_BUF\_FILTER[0].FIL TER | MDR\_CANx-  >CAN\_BUF\_FILTER[x].FILTER Фильтр для приема сообщения в буфер 01 |
| 0x508 | CAN\_BUF\_FILTER[1].MA  SK | MDR\_CANx-  >CAN\_BUF\_FILTER[x].MASK Маска для приема сообщения в буфер 02 |
|  | … |  |
| 0x5FC | CAN\_BUF\_FILTER[31].FI LTER | MDR\_CANx-  >CAN\_BUF\_FILTER[x].FILTER Фильтр для приема сообщения в буфер 32 |

#### 1.2.2.5 Описание библиотечных функций контроллера CAN

Для работы с CAN Миландром предусмотрена специальная библиотека, которая отвечает за работу модуля в микроконтроллере. В листинге 2 представлены функции и структуры данной библиотеки, используемые для работы с контроллером CAN в МК в данном курсовом проекте, с комментариями на русском языке.

Листинг 2 – Основные функции и структуры библиотеки CAN

/\*\*

\* @brief Структура CAN

\*/

typedef struct

{

uint8\_t CAN\_ROP; /\*!< Получение собственных пакетов \*/

uint8\_t CAN\_SAP; /\*!< Отправка ACK на собственные пакеты \*/

uint8\_t CAN\_STM; /\*!< Включение Self Test Mode \*/

uint8\_t CAN\_ROM; /\*!< Включение Read Only Mode \*/

uint32\_t CAN\_PSEG; /\*!< Время распростанения. \*/

uint32\_t CAN\_SEG1; /\*!< Время Phase Segment 1 \*/

uint32\_t CAN\_SEG2; /\*!< Время the Phase Segment 2 \*/

uint32\_t CAN\_SJW; /\*!< Время синхронизации \*/

uint32\_t CAN\_SB; /\*!< Установка Sampling Mode. \*/

uint16\_t CAN\_BRP; /\*!< Установка предделителя по следующему алгоритму:

CLK = PCLK/(BRP + 1)

TQ(microsec) = (BRP + 1)/CLK(MHz) \*/

uint8\_t CAN\_OVER\_ERROR\_MAX; /\*!< Число максимально возможных ошибок на линии. \*/

}CAN\_InitTypeDef;

/\*\*

\* @brief CAN Buffer Data container definition

\*/

typedef uint32\_t CAN\_DataTypeDef[2];

/\*\*

\* @brief Структура передаваемого сообщения

\*/

typedef struct

{

uint32\_t ID; /\*!< Полный идентификатор устройства

Может быть от 0 до 0x1FFFFFFF. \*/

uint8\_t PRIOR\_0; /\*!< Приоритет сообщения \*/

uint8\_t IDE; /\*!< Тип сообщения \*/

uint8\_t DLC; /\*!< Длина кадра

Может быть от 0 до 8 \*/

CAN\_DataTypeDef Data; /\*!< Contains the data to be transmitted. \*/

}CAN\_TxMsgTypeDef;

/\*\*

\* @brief Структура заголовка полученного сообщения

\*/

typedef struct

{

uint32\_t ID; /\*!< Полный идентификатор устройства

Может быть от 0 до 0x1FFFFFFF. \*/

uint8\_t OVER\_EN; /\*!< Может ли полученное сообщение быть перезаписано \*/

uint8\_t IDE; /\*!< Тип сообщения \*/

uint8\_t DLC; /\*!< Длина кадра

Может быть от 0 до 8 \*/

}CAN\_RxMsgHeaderTypeDef;

/\*\*

\* @brief Структура получаемого сообщения

\*/

typedef struct

{

CAN\_RxMsgHeaderTypeDef Rx\_Header; /\*!< Заголовок. \*/

CAN\_DataTypeDef Data; /\*!< Данные\*/

}CAN\_RxMsgTypeDef;

/\*\*

\* @brief Деинициализация всех регистров CAN и установка начальных значений

\* @param CANx: выбор CAN

\* @retval None

\*/

void CAN\_DeInit(MDR\_CAN\_TypeDef\* CANx);

/\*\*

\* @brief Инициализация CAN в соответствии с параметром CAN\_InitStruct.

\* @param CANx: Выбор CAN

\* @param CAN\_InitStruct; Указатель на CAN\_InitTypeDef , где содержится информация о конфигурации CAN

\* @retval None

\*/

void CAN\_Init(MDR\_CAN\_TypeDef\* CANx, const CAN\_InitTypeDef\* CAN\_InitStruct);

/\*\*

\* @brief Заполнение структуры CAN\_InitTypeDef начальными значениями

\* @param CAN\_InitStruct: указатель на CAN\_InitTypeDef

\* @retval None

\*/

void CAN\_StructInit(CAN\_InitTypeDef\* CAN\_InitStruct);

/\*\*

\* @brief Включение и отключение прерываний по CAN

\* @param CANx: Выбор CAN

\* @param CAN\_IT: Выбор прерывния

\* @param NewState: вкл/выкл

\* @retval None.

\*/

void CAN\_ITConfig(MDR\_CAN\_TypeDef\* CANx, uint32\_t CAN\_IT, FunctionalState NewState);

/\*\*

\* @brief Инициализация передачи сообщения

\* @param CANx: Выбор CAN

\* @param BufferNumber: Размер буфера для сообщения

\* @param TxMessage: Структура передаваемого сообщения

\* @retval None

\*/

void CAN\_Transmit(MDR\_CAN\_TypeDef\* CANx, uint32\_t BufferNumber, CAN\_TxMsgTypeDef\* TxMessage);

/\*\*

\* @brief Начало ожидания получения сообщения

\* @param CANx: Выбор CAN

\* @param BufferNumber: Размер буфера для получения сообщения

\* @param OverWrite: Перезапись сообщений вкл/выкл

\* @retval None

\*/

void CAN\_Receive(MDR\_CAN\_TypeDef\* CANx, uint32\_t BufferNumber, FunctionalState OverWrite);

/\*\*

\* @brief Чтение данных из буфера

\* @param CANx: Выбор CAN

\* @param BufferNumber: Размер буфера для получения сообщения

\* @param RxBuffer: Массив структур CAN\_DataTypeDef для получения сообщения

\* @retval None

\*/

void CAN\_GetReceivedData(MDR\_CAN\_TypeDef\* CANx, uint32\_t BufferNumber, CAN\_DataTypeDef RxBuffer);

/\*\*

\* @brief Контроллер CAN вкл/выкл

\* @param CANx: Выбор CAN.

\* @param NewState: вкл/выкл

\* @retval None

\*/

void CAN\_Cmd(MDR\_CAN\_TypeDef\* CANx, FunctionalState NewState);

/\*\*

\* @brief Возвращает статус регистра CAN

\* @param CANx: Выбор CAN

\* @retval Статус регистра CAN

\*/

uint32\_t CAN\_GetStatus(MDR\_CAN\_TypeDef\* CANx);

/\*\*

\* @brief Проверка флага прерывания

\* @param CANx: Выбор CAN

\* @param CAN\_IT: Тип прерывания

\* @retval установлен/не установлен

\*/

ITStatus CAN\_GetITState(MDR\_CAN\_TypeDef\* CANx, uint32\_t CAN\_IT);

/\*\*

\* @brief Разрешение прерывания по получению сообщения

\* @param CANx: Выбор CAN

\* @param Buffer\_IT: Размер буфера по переполнению которого будет вызвано перерывание

\* @param NewState: вкл/выкл

\* @retval None.

\*/

void CAN\_RxITConfig(MDR\_CAN\_TypeDef\* CANx, uint32\_t Buffer\_IT, FunctionalState NewState);

/\*\*

\* @brief Проверка флага прерывания CAN по получению сообщения

\* @param CANx: Выбор CAN

\* @param BufferNumber: Размер буфера

\* @retval установлен/не установлен

\*/

ITStatus CAN\_GetRxITStatus(MDR\_CAN\_TypeDef\* CANx, uint32\_t BufferNumber);

/\*\*

\* @brief Очистка флагов прерывания CAN

\* @param CANx: Выбор CAN

\* @param BufferNumber: Размер буфера

\* @param Status\_Flag: Выбор флага прерывания для очистки

\* @retval None.

\*/

void CAN\_ITClearRxTxPendingBit(MDR\_CAN\_TypeDef\* CANx, uint32\_t BufferNumber, uint32\_t Status\_Flag);

### 1.2.3 SSP

#### 1.2.3.1 Общая информация о модуле

Модуль порта синхронной последовательной связи (SSP – Synchronous Serial Port) выполняет функции интерфейса последовательной синхронной связи в режиме ведущего и ведомого устройства и обеспечивает обмен данными с подключенным ведомым или ведущим периферийным устройством в соответствии с одним из протоколов:

* интерфейс SPI фирмы Motorola;
* интерфейс SSI фирмы Texas Instruments;
* интерфейс Microwire фирмы National Semiconductor.

Как в ведущем, так и в ведомом режиме работы модуль SSP обеспечивает:

* преобразование данных, размещенных во внутреннем буфере FIFO передатчика (восемь 16-разрядных ячеек данных) из параллельного в последовательный формат;
* преобразование данных из последовательного в параллельный формат и их запись в аналогичный буфер FIFO приемника (восемь 16-разрядный ячеек данных).

Модуль формирует сигналы прерываний по следующим событиям:

* необходимость обслуживания буферов FIFO приемника и передатчика;
* переполнение буфера FIFO приемника;
* наличие данных в буфере FIFO приемника по истечении времени таймаута.

Основные сведения о модуле представлены в следующих разделах:

* характеристики интерфейса SPI;
* характеристики интерфейса Microwire;

Так как для тестирования модуля достаточно одного из трех интерфейсов, то рассмотрим работу только с интерфейсом SPI фирмы Motorolla. Данный интерфейс был выбран как наиболее используемый среди остальных. Кроме того, данный интерфейс был изучен в ходе курса «Микропроцессорные системы», в связи с чем не нуждается в дополнительном изучении.

#### 1.2.3.2 Прием и передача данных

Модуль SSP содержит программируемые делители частоты, формирующие тактовый сигнал обмена данными SSP\_CLK из сигнала, поступающего на линию SSPCLK. Скорость передачи данных может достигать более 2 МГц, в зависимости от частоты SSPCLK и характеристик подключенного периферийного устройства.

Режим обмена данными, формат информационного кадра и количество бит данных задаются программно с помощью регистров управления CR0 и CR1.

Модуль формирует четыре независимо маскируемых прерывания:

SSPTXINTR – запрос на обслуживание буфера передатчика;

SSPRXINTR – запрос на обслуживание буфера приемника;

SSPRORINTR – переполнение приемного буфера FIFO;

SSPRTINTR – таймаут ожидания чтения данных из приемного FIFO.

Кроме того, формируется общий сигнал прерывания SSPINTR, возникающий в случае активности одного из вышеуказанных независимых немаскированных прерываний, который идет на контроллер NVIC.

Модуль также формирует сигналы запроса на прямой доступ к памяти (DMA) для совместной работы с контроллером DMA.

В зависимости от режима работы модуля сигнал SSPFSSOUT используется для выбора ведомого режима (активное состояние – низкий уровень).

Буфер передатчика имеет ширину 16 бит, глубину 8 слов, схему организации доступа типа FIFO – «первый вошел, первый вышел». Данные от центрального процессора сохраняются в буфере до тех пор, пока не будут считаны блоком передачи данных.

Буфер приемника имеет ширину 16 бит, глубину 8 слов, схему организации доступа типа FIFO – «первый вошел, первый вышел». Принятые от периферийного устройства данные сохраняются в этом буфере блоком приема данных в до тех пор, пока не будут считаны центральным процессором.

В режиме ведущего устройства модуль формирует тактовый сигнал обмена данными SSP\_CLK для подключенных ведомых устройств. Как было описано ранее, данный сигнал формируется путем деления частоты сигнала SSPCLK.

Блок передатчика последовательно считывает данные из буфера FIFO передатчика и производит их преобразование из параллельной формы в последовательную. Далее поток последовательных данных и элементов кадровой синхронизации, тактированный сигналом SSP\_CLK, передаётся по линии SSP\_TXD к подключенным ведомым устройствам.

Блок приемника выполняет преобразование данных, поступающих синхронно с линии SSP\_RXD, из последовательной в параллельную форму, после чего загружает их в буфер FIFO приемника, откуда они могут быть считаны процессором.

В режиме ведомого устройства тактовый сигнал обмена данными формируется одним из подключенных к модулю периферийных устройств и поступает по линии SSP\_CLK.

При этом блок передатчика, тактируемый этим внешним сигналом, считывает данные из буфера FIFO, преобразует их из параллельной формы в последовательную, после чего выдает поток последовательных данных и элементов кадровой синхронизации в линию SSP\_TXD.

Аналогично, блок приемника выполняет преобразование данных, поступающих с линии SSP\_RXD синхронно с сигналом SSP\_CLK, из последовательной в параллельную форму, после чего загружает их в буфер FIFO приемника, откуда они могут быть считаны процессором.

#### 1.2.3.3 Описание регистров контроллера SSP

Описание регистров с базовым адресом, их названием и описанием контроллера SSP представлено в таблице 6. Отдельно рассмотрим два управляющих регистра. Они представлены в таблицах 7 и 8 соответственно.

Таблица 6 – Описание регистров контроллера SSP

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Базовый адрес** | **Наименование** | **Тип** | **Значение после сброса** | **Размер, бит** | **Описание** |
| 0x4004\_0000 | MDR\_SSP1 |  |  |  | Контроллер SSP1 |
| 0x400A\_0000 | MDR\_SSP2 |  |  |  | Контроллер SSP2 |
| **Смещение** |  |  |  |  |  |
| 0x000 | CR0 | RW | 0x0000 | 16 | Регистр MDR\_SSPx->CR0 управления 0 |
| 0x004 | CR1 | RW | 0x0 | 4 | Регистр MDR\_SSPx->CR1 управления 1 |
| 0x008 | DR | RW | 0x---- | 16 | Буфера FIFO приемника (чтение) Буфер FIFO передатчика (запись) MDR\_SSPx->DR |
| 0x00С | SR | RO | 0x03 | 3 | Регистр MDR\_SSPx->SR состояния |
| 0x010 | CPSR | RW | 0x00 | 8 | Регистр  MDR\_SSPx->CPSR делителя тактовой частоты |
| 0x014 | IMSC | RW | 0x0 | 4 | Регистр MDR\_SSPx->IMSC маски прерывания |
| 0x018 | RIS | RO | 0x8 | 4 | Регистр  MDR\_SSPx->RIS состояния прерываний без учета маскирования |
| 0x01C | MIS | RO | 0x0 | 4 | Регистр  MDR\_SSPx->MIS состояния прерываний с учетом маскирования |
| 0x020 | ICR | WO | 0x0 | 4 | Регистр  MDR\_SSPx->ICR сброса прерывания |
| 0x024 | DMACR | RW | 0x0 | 2 | Регистр MDR\_SSPx->DMACR управления прямым доступом к памяти |

Таблица 7 – Описание управляющего регистра CR0 контроллера SSP

|  |  |  |
| --- | --- | --- |
| **№**  **бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
| 31…16 | - | Зарезервировано |
| 15…8 | SCR | Скорость последовательного обмена.  Значение поля SCR используется при формировании тактового сигнала обмена данными. Информационная скорость удовлетворяет соотношению: F\_SSPCLK / (CPSDVR \* (1 + SCR)), где CPSDVR – четное число в диапазоне от 2 до 254 (см.  регистр SSPCPSR), а SCR – число от 0 до 255 |
| 7 | SPH | Фаза сигнала SSPCLKOUT (используется только в режиме обмена SPI фирмы Motorola). См. раздел «Формат SPI фирмы Motorola» |
| 6 | SPO | Полярность сигнала SSPCLKOUT (используется только в режиме обмена SPI фирмы Motorola). См. раздел «Формат синхронного обмена SPI фирмы Motorola» |
| 5…4 | FRF | Формат информационного кадра.  00 - протокол SPI фирмы Motorola;  01 - протокол SSI фирмы Texas Instruments;  10 - протокол Microwire фирмы National Semiconductor;  11 - резерв |
| 3…0 | DSS | Размер слова данных:  0000 – 0010 : резерв  0011 – 1111 : 4 бита – 16 бит |

Таблица 8 – Описание управляющего регистра CR1 контроллера SSP

|  |  |  |
| --- | --- | --- |
| **№**  **бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
| 15…4 |  | Резерв, при чтении результат не определен. При записи следует устанавливать в 0 |

Окончание таблицы 8

|  |  |  |
| --- | --- | --- |
| 3 | SOD | Запрет выходных линий в режиме ведомого устройства.  Бит используется только в режиме ведомого устройства (MS=1). Это позволяет организовать двусторонний обмен данными в системах, содержащих одно ведущее и несколько ведомых устройств.  Бит SOD следует установить в случае, если данный ведомый модуль SSP не должен в настоящее время осуществлять передачу данных в линию SSP\_TXD. При этом линии обмена данных ведомых устройств можно соединить параллельно.   1. – управление линией SSP\_TXD в ведомом режиме разрешена. 2. – управление линией SSP\_TXD в ведомом режиме запрещена |
| 2 | MS | Выбор ведущего или ведомого режима работы:  0 – ведущий модуль (устанавливается по умолчанию); 1 – ведомый модуль |
| 1 | SSE | Разрешение работы приемопередатчика:   1. – работа запрещена; 2. – работа разрешена |
| 0 | LBM | Тестирование по шлейфу:   1. – нормальный режим работы приемопередатчика; 2. – выход регистра сдвига передатчика соединен со входом регистра сдвига приемника |

#### 1.2.3.4 Описание библиотечных функций контроллера SSP

Для работы с SSP Миландром предусмотрена специальная библиотека, которая отвечает за работу модуля в микроконтроллере. В листинге 3 представлены функции и структуры данной библиотеки, используемые для работы с контроллером CAN в МК в данном курсовом проекте, с комментариями на русском языке.

Листинг 3 – Основные функции и структуры библиотеки SSP

/\*\*

\* @brief Структура SSP

\*/

typedef struct

{

uint16\_t SSP\_SCR; /\*!< Скорость SSP. Определяется по формуре:

F\_SSPCLK / ( CPSDVR \* (1 + SCR) ) \*/

uint16\_t SSP\_CPSDVSR; /\*!< Конфигурация SSP clock divider. Может быть от 2 да 254 \*/

uint16\_t SSP\_Mode; /\*!< Тип SSP \*/

uint16\_t SSP\_WordLength; /\*!< Число битов получаемых и передаваемых в одном кадре \*/

uint16\_t SSP\_SPH; /\*!< Фаза передачи \*/

uint16\_t SSP\_SPO; /\*!< Полярность \*/

uint16\_t SSP\_FRF; /\*!< Формат кадра \*/

uint16\_t SSP\_HardwareFlowControl; /\*!< определяет включен или выключен HardwareFlowControl \*/

}SSP\_InitTypeDef;

/\*\*

\* @brief Перезагрузка SSP к начальным значениям

\* @param SSPx: Выбор SSP

\* @retval None

\*/

void SSP\_DeInit(MDR\_SSP\_TypeDef\* SSPx);

/\*\*

\* @brief Инициализирует SSP по структуре из параметров

\* @param SSPx: Выбор SSP

\* @param SSP\_InitStruct: указатель на SSP\_InitTypeDef structure

\* @retval None

\*/

void SSP\_Init(MDR\_SSP\_TypeDef\* SSPx, const SSP\_InitTypeDef\* SSP\_InitStruct);

/\*\*

\* @brief Заполнение каждого члена структуры SSP\_InitStruct начальным значением

\* @param SSP\_InitStruct: указатель на SSP\_InitTypeDef structure

\* @retval None

\*/

void SSP\_StructInit(SSP\_InitTypeDef\* SSP\_InitStruct);

/\*\*

\* @brief вкл/выкл SSP

\* @param SSPx: Выбор SSP

\* @param NewState: вкл/выкл

\* @retval None

\*/

void SSP\_Cmd(MDR\_SSP\_TypeDef\* SSPx, FunctionalState NewState);

/\*\*

\* @brief Разрешение прерываний по SSP

\* @param SSPx: Выбор SSP

\* @param SSP\_IT: Тип прерывания. Может быть

\* @arg SSP\_IT\_TX

\* @arg SSP\_IT\_RX

\* @arg SSP\_IT\_RT

\* @arg SSP\_IT\_ROR

\* @param NewState: вкл/выкл

\* @retval None

\*/

void SSP\_ITConfig(MDR\_SSP\_TypeDef\* SSPx, uint32\_t SSP\_IT, FunctionalState NewState);

/\*\*

\* @brief Проверяет произошло ли прерывание

\* @param SSPx: Выбор SSP

\* @param SSP\_IT: Тип прерывания. Может быть:

\* @arg SSP\_IT\_TX

\* @arg SSP\_IT\_RX

\* @arg SSP\_IT\_RT

\* @arg SSP\_IT\_ROR

\* @retval установлен/не установлен

\*/

ITStatus SSP\_GetITStatus(MDR\_SSP\_TypeDef\* SSPx, uint32\_t SSP\_IT);

/\*\*

\* @brief Очистка битов прерывания SSP

\* @param SSPx: Выбор SSP

\* @param SSP\_IT: Выбор бита для очистки. Может быть:

\* @arg SSP\_IT\_RT

\* @arg SSP\_IT\_ROR

\* @retval None

\*/

void SSP\_ClearITPendingBit(MDR\_SSP\_TypeDef\* SSPx, uint32\_t SSP\_IT);

/\*\*

\* @brief Предача данных через SSP

\* @param SSPx:Выбор SSP

\* @param Data: Данные для передачи

\* @retval None

\*/

void SSP\_SendData(MDR\_SSP\_TypeDef\* SSPx, uint16\_t Data);

/\*\*

\* @brief Возвращает последние полученные данные по SSP

\* @param SSPx: Выбор SSP

\* @retval Полученные данные (7:0) and флаги ошибки(15:8).

\*/

uint16\_t SSP\_ReceiveData(MDR\_SSP\_TypeDef\* SSPx);

/\*\*

\* @brief Инициализация предделителя для SSP

\* @param SSPx: Выбор SSP

\* @param SSP\_BRG: Предделитель. Может быть:

\* @arg SSP\_HCLKdiv1

\* @arg SSP\_HCLKdiv2

\* @arg SSP\_HCLKdiv4

\* @arg SSP\_HCLKdiv8

\* @arg SSP\_HCLKdiv16

\* @arg SSP\_HCLKdiv32

\* @arg SSP\_HCLKdiv64

\* @arg SSP\_HCLKdiv128

\* @retval None

\*/

void SSP\_BRGInit(MDR\_SSP\_TypeDef\* SSPx, uint32\_t SSP\_BRG);

### 1.2.4 UART

#### 1.2.4.1 Общее описание модуля

Модуль универсального асинхронного приемопередатчика (UART – Universal Asynchronous Receiver-Transmitter) представляет собой периферийное устройство микроконтроллера.

В состав контроллера включен кодек (ENDEC – ENcoder/DEcoder) последовательного интерфейса инфракрасной (ИК) передачи данных в соответствии с протоколом SIR (SIR – Serial Infra Red) ассоциации Infrared Data Association (IrDA).

Так как интерфейс UART, аналогично интерфейсу SPI был изучен ранее, его дополнительное описание не требуется.

#### 1.2.4.2 Основные характеристики модуля UART

Может быть запрограммирован для использования, как в качестве универсального асинхронного приемопередатчика, так и для инфракрасного обмена данными (SIR).

Содержит независимые буферы приема (16x12) и передачи (16x8) типа FIFO (First In First Out – первый вошел, первый вышел), что позволяет снизить интенсивность прерываний центрального процессора.

Программное отключение FIFO позволяет ограничить размер буфера одним байтом.

Программное управление скоростью обмена. Обеспечивается возможность деления тактовой частоты опорного генератора в диапазоне (1x16 – 65535x16). Допускается использование нецелых коэффициентов деления частоты, что позволяет использовать любой опорный генератор с частотой более 3.6864 МГц.

Поддержка стандартных элементов асинхронного протокола связи – стартового и стопового бит, а та же бита контроля четности, которые добавляются перед передачей и удаляются после приема.

Независимое маскирование прерываний от буфера FIFO передатчика, буфера FIFO приемника, по таймауту приемника, по изменению линий состояния модема, а также в случае обнаружения ошибки.

Поддержка прямого доступа к памяти.

Обнаружение ложных стартовых бит.

Формирование и обнаружения сигнала разрыва линии.

Поддержка функция управления модемом (линии CTS, DCD, DSR, RTS, DTR и RI).

Возможность организации аппаратного управления потоком данных.

Полностью программируемый асинхронный последовательный интерфейс с характеристиками:

* данные длиной 5, 6, 7 или 8 бит;
* формирование и контроль четности (проверочный бит выставляется по четности, нечетности, имеет фиксированное значение, либо не передается);
* формирование 1 или 2 стоповых бит;
* скорость передачи данных – от 0 до UARTCLK/16 Бод.

#### 1.2.4.3 Описание регистров контроллера UART

Описание регистров с базовым адресом, их названием и описанием контроллера UART представлено в таблице 9.

Таблица 9 – Описание регистров контроллера UART

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Смещение** | **Наименование** | **Тип** | **Значение после сброса** | **Размер, бит** | **Описание** |
| 0x40030000 | MDR\_UART1 |  |  |  | Контроллер UART1 |
| 0x40038000 | MDR\_UART2 |  |  |  | Контроллер UART2 |
| 0x000 | DR | RW | 0x--- | 12/8 | MDR\_UARTx->DR  Регистр данных |
| 0x004 | RSR\_ECR | RW | 0x0 | 4/0 | MDR\_UARTx->RSR\_ECR  Регистр состояния приемника / Сброс ошибки приемника |
| 0x008- 0x014 |  |  |  |  | Зарезервировано |
| 0x018 | FR | RO | 0b-10010--- | 9 | MDR\_UARTx->FR  Регистр флагов |
| 0x01C |  |  |  |  | Зарезервировано |
| 0x020 | ILPR | RW | 0x00 | 8 | MDR\_UARTx->ILPR  Регистр управления ИК обменом в режиме пониженного энергопотребления |
| 0x024 | IBRD | RW | 0x0000 | 16 | MDR\_UARTx->IBRD  Целая часть делителя скорости обмена данными |
| 0x028 | FBRD | RW | 0x00 | 6 | MDR\_UARTx->FBRD  Дробная часть делителя скорости обмена данными |
| 0x02C | LCR\_H | RW | 0x00 | 8 | MDR\_UARTx->LCR\_H  Регистр управления линией |
| 0x030 | CR | RW | 0x0300 | 16 | MDR\_UARTx->CR  Регистр управления |
| 0x034 | IFLS | RW | 0x12 | 6 | MDR\_UARTx->IFLS  Регистр порога прерывания по заполнению буфера FIFO |
| 0x038 | IMSC | RW | 0x000 | 11 | MDR\_UARTx->IMSC  Регистр маски прерывания |

Окончание таблицы 9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0x03C | RIS | RO | 0x00- | 11 | MDR\_UARTx->RIS  Регистр состояния прерываний |
| 0x040 | MIS | RO | 0x00- | 11 | MDR\_UARTx->MIS  Регистр состояния прерываний с маскированием |
| 0x044 | ICR | WO | - | 11 | MDR\_UARTx->ICR  Регистр сброса прерывания |
| 0x048 | DMACR | RW | 0x00 | 3 | MDR\_UARTx->DMACR  Регистр управления DMA |

#### 1.2.4.4 Описание библиотечных функций контроллера UART

Для работы с UART Миландром предусмотрена специальная библиотека, которая отвечает за работу модуля в микроконтроллере. В листинге 4 представлены функции и структуры данной библиотеки, используемые для работы с контроллером UART в МК в данном курсовом проекте, с комментариями на русском языке.

Листинг 4 – Основные функции и структуры библиотеки UART

/\*\*

\* @brief Структура UART

\*/

typedef struct

{

uint32\_t UART\_BaudRate; /\*!< Скорость UART. Определяется по формуле

- IntegerDivider = ((UARTCLK) / (16 \* (UART\_InitStruct->UART\_BaudRate)))

- FractionalDivider = ((IntegerDivider - ((u32) IntegerDivider)) \* 64) + 0.5 \*/

uint16\_t UART\_WordLength; /\*!< Длина слова \*/

uint16\_t UART\_StopBits; /\*!< Количество стопбит \*/

uint16\_t UART\_Parity; /\*!< Четность. \*/

uint16\_t UART\_FIFOMode; /\*!< Вкл/выкл FIFOMode \*/

uint16\_t UART\_HardwareFlowControl; /\*!< Определяет включение HardwareFlowControl \*/

}UART\_InitTypeDef;

/\*\*

\* @brief Перезагрузка UART к начальным значениям

\* @param UARTx: Выбор UART

\* @retval None

\*/

void UART\_DeInit(MDR\_UART\_TypeDef\* UARTx);

/\*\*

\* @brief Инициализирует UART по структуре из параметров

\* @param UARTx: Выбор UART

\* @param UART\_InitStruct: указатель на структуру UART\_InitTypeDef

\* @retval Статус скорости передачи

\*/

BaudRateStatus UART\_Init(MDR\_UART\_TypeDef\* UARTx, UART\_InitTypeDef\* UART\_InitStruct);

/\*\*

\* @brief Заполнение всех значений UART\_InitStruct начальными значениями

\* @param UART\_InitStruct: указатель на структуру UART\_InitTypeDef

\* @retval None

\*/

void UART\_StructInit(UART\_InitTypeDef\* UART\_InitStruct);

/\*\*

\* @brief Включение контроллера UART

\* @param UARTx: Выбор UART

\* @param NewState: вкл/выкл

\* @retval None

\*/

void UART\_Cmd(MDR\_UART\_TypeDef\* UARTx, FunctionalState NewState);

/\*\*

\* @brief Разрешение прерываний по UART

\* @param UARTx: Выбор UART

\* @param UART\_IT: Тип прерывания.

\* @param NewState: вкл/выкл

\* @retval None

\*/

void UART\_ITConfig(MDR\_UART\_TypeDef\* UARTx, uint32\_t UART\_IT, FunctionalState NewState);

/\*\*

\* @brief Проверка наступления прерывания

\* @param UARTx: Выбор UART

\* @param UART\_IT: Тип прерывания

\* @retval установлен/не установлен

\*/

ITStatus UART\_GetITStatus(MDR\_UART\_TypeDef\* UARTx, uint32\_t UART\_IT);

/\*\*

\* @brief Очистка битов прерывания UART

\* @param UARTx: Выбор UART

\* @param UART\_IT: Тип прерывания

\* @retval None

\*/

void UART\_ClearITPendingBit(MDR\_UART\_TypeDef\* UARTx, uint32\_t UART\_IT);

/\*\*

\* @brief Отправка данных по UART

\* @param UARTx: Выбор UART

\* @param Data: Данные для отправки

\* @retval None

\*/

void UART\_SendData(MDR\_UART\_TypeDef\* UARTx, uint16\_t Data);

/\*\*

\* @brief Возвращает последние полученные данные по UART

\* @param UARTx: Выбор UART

\* @retval Полученные данные (7:0) и флаги ошибок (15:8).

\*/

uint16\_t UART\_ReceiveData(MDR\_UART\_TypeDef\* UARTx);

/\*\*

\* @brief Настройка предделителя UART

\* @param UARTx: Выбор UART

\* @param UART\_BRG: Установка предделителя

\* @retval None

\*/

void UART\_BRGInit(MDR\_UART\_TypeDef\* UARTx, uint32\_t UART\_BRG);

## 1.3 Описание отладочной платы

Для отладки курсового проекта использовалось две отладочных платы для микросхемы К1986ВЕ92QI (ТСКЯ.469575.002-01 вер. 4). Схема расположения элементов на данной отладочной плате представлена на рисунке 9.

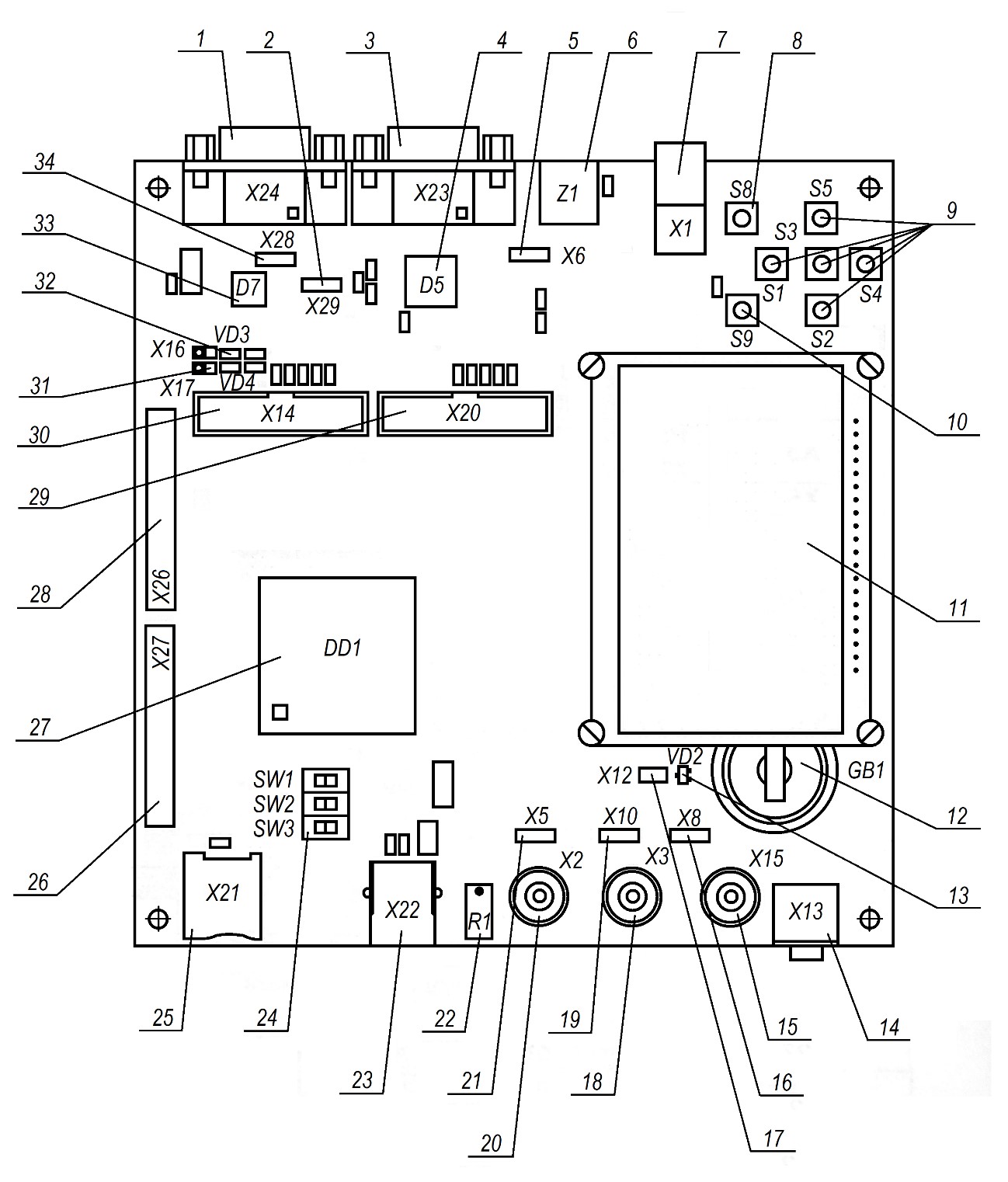


Рисунок 9 - Расположение элементов управления и коммутации отладочной платы

Кнопки S1 «UP», S2 «LEFT», S3 «SELECT», S4 «DOWN», S5 «RIGHT» могут быть нами запрограммированы. Кнопка S1 подключена к линии PB5 порта B, кнопка S2 – к линии PE3 порта E, кнопка S3 – к линии PC2 порта C, кнопка S4 – к линии PE1 порта E, кнопка S5 – к линии PB6 порта B.

Кнопка S8 «RESET» предназначена для аппаратного сброса.

Кнопка S9 «WAKEUP» служит для выхода микроконтроллера из режима пониженного энергопотребления STANDBY.

Светодиоды VD3 и VD4 (поз. 32 на рисунке 1.3) подключены через ограничивающие ток резисторы к линиям PC0 и PC1 порта C и могут служить для простейшей индикации.

Описание элементов отладочной платы сведено в таблицу 10.

Таблица 10 – элементы отладочной платы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обозначение** | **Описание** | | **Поз.** | |
| DD1 | Контактное устройство для микроконтроллера | | 27 | |
| D5 | Приемопередатчик RS-232 | | 4 | |
| D7 | Приемопередатчик CAN | | 33 | |
| GB1 | Батарейный отсек | | 12 | |
| R1 | Подстроечный резистор канала 7 АЦП | | 22 | |
| SW1SW3 | Переключатели | | 24 | |
| S1-S5 | Кнопки UP, LEFT, SELECT, DOWN, RIGHT | | 9 | |
| S8 | Кнопка RESET | | 8 | |
| S9 | Кнопка WAKEUP | | 10 | |
| VD2 | Транзистор для подключения батарейного отсека | | 13 | |
| VD3, VD4 | Набор светодиодов для порта С | | 32 | |
| X1 | Разъем питания 5В | | 7 | |
| X2 | Разъем BNC внешнего сигнала канала 7 АЦП | | 20 | |
| X3 | Разъем BNC внешнего сигнала на 1-м входе компаратора | | 18 | |
| X5 | | Разъем для установки конфигурационных перемычек | | 21 |
| X6 | | 5 |
| X8 | | 16 |
| X10 | | 19 |
| X12 | | 17 |

|  |  |  |
| --- | --- | --- |
| X13 | Разъем Audio 3,5 мм выхода ЦАП1 через звуковой усилитель | 14 |
| X14 | Разъем отладки JTAG-A | 30 |
| X15 | Разъем BNC выхода ЦАП-1 | 15 |
| X16, X17 | Разъемы для установки конфигурационных перемычек | 31 |
| X20 | Разъем отладки JTAG-B | 29 |
| X21 | Разъем карты памяти micro-SD | 25 |

Окончание таблицы 10

|  |  |  |
| --- | --- | --- |
| X22 | Разъем USB-B | 23 |
| X23 | Разъем интерфейса RS-232 | 3 |
| X24 | Разъем интерфейса CAN | 1 |
| X26 | Разъем портов B, C, D микроконтроллера | 28 |
| X27 | Разъем портов A, E, F микроконтроллера | 26 |
| X28 | Разъем для установки конфигурационных перемычек | 34 |
| X29 | 2 |
| Z1 | Фильтр питания | 6 |
| – | Жидкокристаллический модуль | 11 |

Отдельно рассмотрим LCD и элементы, используемые для работы с интерфейсами, то есть модули: приемопередатчик CAN и приемопередатчик RS-232.

### 1.3.1 LCD

Модуль LCD представлен на плате в виде MT– 12864J v.1 отечественной компании «МЭЛТ» содержит собственный контроллер управления и жидкокристаллическую панель. Модуль содержит оперативное запоминающее устройство (ОЗУ) для хранения данных, выводимых на экран, размером 64х64х2 бит. Каждой светящейся точке на экране соответствует логическая «1» в ячейке ОЗУ модуля.

Функциональная блок-схема данного модуля представлена на рисунке 10.

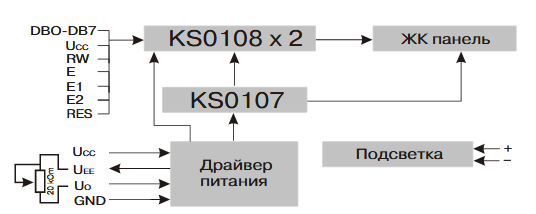


Рисунок 10 – Функциональная блок-схема MT– 12864J

В таблице 11 представлены назначения внешних выводов LCD.

Таблица 11 – Назначение внешних выводов LCD

|  |  |  |
| --- | --- | --- |
| **№** | **Обозначение** | **Назначение** |
| 1 | UCC | Напряжение питания |
| 2 | GND | Общий вывод |
| 3 | U0 | Управление контрастностью |
| 4 - 11 | DB0 – DB7 | Шина данных |

Окончание таблицы 11

|  |  |  |
| --- | --- | --- |
| 12 | E1 | Выбор кристалла 1 |
| 13 | E2 | Выбор кристалла 2 |
| 14 | RES | Сброс (начальная установка) |
| 15 | R/W | Выбор: Чтение/ Запись |
| 16 | A0 | Выбор: Команды/ Данные |
| 17 | E | Стробирование данных |
| 18 | UEE | Выход DC–DC преобразователя |
| 19 | А | + питания подсветки |
| 20 | K | – питания подсветки |

Все команды с расшифровками, используемые в работе с данным LCD представлены на рисунке 11.

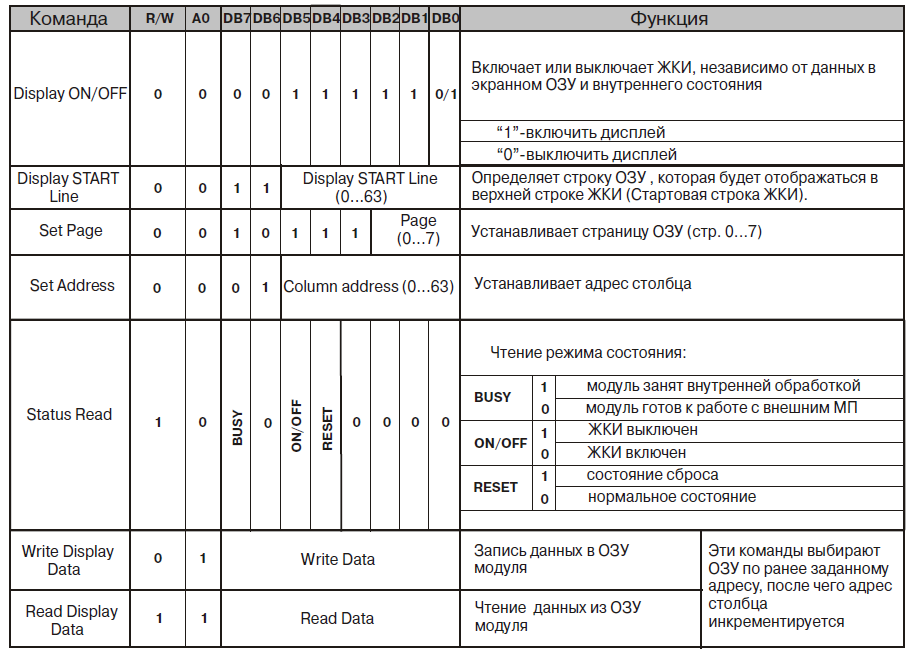


Рисунок 11 – Команды для работы с LCD

### 1.3.2 Приемопередатчик CAN

Приемопередатчик CAN представлен на данной отладочной плате в виде модуля ATA6660.

ATA6660 - ИС, изготовленная по технологии Smart Power BCD60-III компании Atmel. Она специально разработана для обеспечения высокоскоростной передачи данных по дифференциальным линиям в жестких климатических условиях, например, в автомобильном или индустриальном оборудовании. Приемопередатчик имеет пропускную способность 1 МБод. ATA6660 полностью совместим с ISO11898, стандартом для высокоскоростных коммуникационных CAN-C.

Описание выводов данной ИС представлено в таблице 12.

Таблица 12 – Выводы ATA6660

|  |  |  |
| --- | --- | --- |
| Вывод | Обозначение | Функция |
| 1 | TXD | Вход передатчика данных |
| 2 | GND | Общий |
| 3 | VCC | Напряжение питания |
| 4 | RXD | Выход приемника данных |
| 5 | VREF | Выход источника опорного напряжения |
| 6 | CANL | Вход/выход CAN-приемопередатчика (низкий уровень) |
| 7 | CANH | Вход/выход CAN-приемопередатчика (высокий уровень) |
| 8 | RS | Управление: дежурный режим/рабочий режим |

Временные диаграммы данного приемопередатчика представлены на рисунке 12.

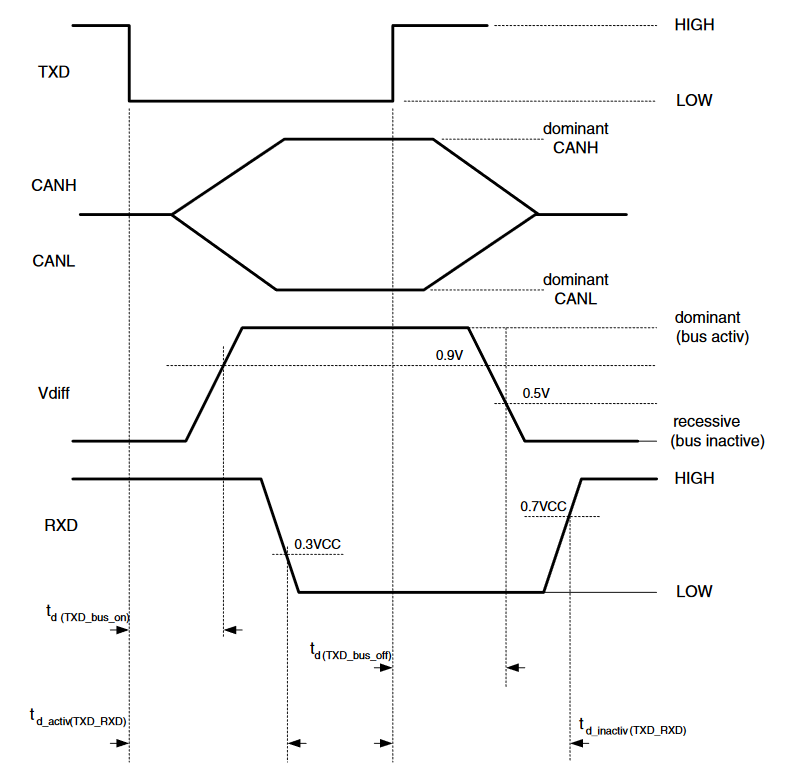


Рисунок 12 – Временные диаграммы приемопередатчика CAN

На рисунке 13 представлена схема подключения рассматриваемого модуля в шину CAN.

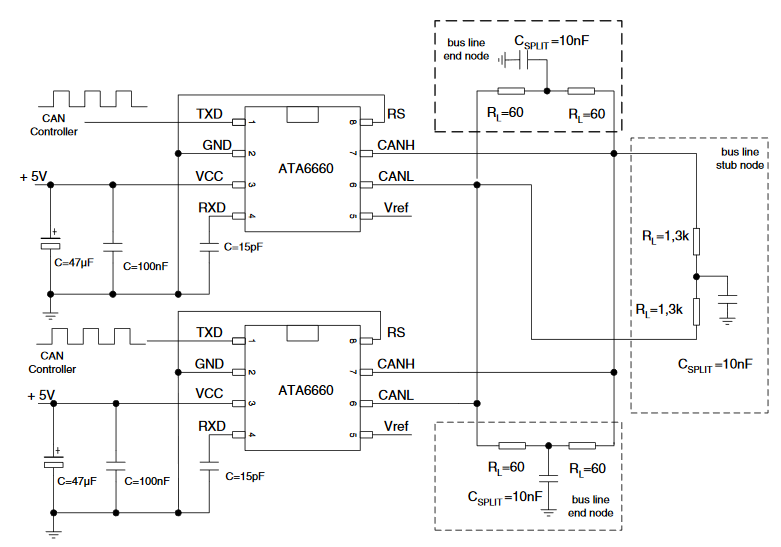


Рисунок 13 – Схема подключения модуля приемопередатчика CAN

### 1.3.3 Приемопередатчик RS-232

Приемопередатчик RS-232 представлен в виде модуля 555ИН4 производства Миландр.

Приемопередатчик интерфейса RS-232 содержит пять передатчиков КМОП –RS-232 и три приемника RS-232 –КМОП (один активный во всех режимах), а также внутренний импульсный преобразователь напряжения с внешними конденсаторами. Для работы схемы требуется 4 внешних конденсатора. Особенность схемы – наличие режима “выключено”, в котором все приемники остаются активными. В этом режиме ток потребления составляет не более 10мкА.

Описание выводов данной микросхемы представлено в таблице 13.

Таблица 13 – Выводы 555ИН4

|  |  |  |  |
| --- | --- | --- | --- |
| Вывод корпуса | Контактная площадка кристалла | Условное обозначение | Функциональное назначение выводов |
| 1 | 1 | С2+ | Положительный вывод конденсатора для внутреннего импульсного преобразователя напряжения |

Окончание таблицы 13

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 2 | GND | Общий |
| 3 | 3 | С2- | Отрицательный вывод конденсатора для внутреннего импульсного преобразователя напряжения |
| 4 | 4 | UL | -5,5 В вывод внутреннего импульсного преобразователя напряжения |
| 5 | 5 | Т1OUT | Выход передатчика RS-232 |
| 6 | 6 | T2OUT | Выход передатчика RS-232 |
| 7 | 7 | T3OUT | Выход передатчика RS-232 |
| 8 | 8 | T4OUT | Выход передатчика RS-232 |
| 9 | 9 | T5OUT | Выход передатчика RS-232 |
| 10 | 10 | T5IN | Вход передатчика RS-232 |
| 11 | 11 | T4IN | Вход передатчика RS-232 |
| 12 | 12 | T3IN | Вход передатчика RS-232 |
| 13 | 13 | T2IN | Вход передатчика RS-232 |
| 14 | 14 | T1IN | Вход передатчика RS-232 |
| 15 | 15 | MBAUD | Вход управления режимом передачи  250/1000 Кбит/с  Активный уровень “1” |
| 16 | 16 | nNSHDN | Вход выключения передатчиков RS-232. Активный уровень “0” |
| 17 | 17 | nEN | Вход разрешения работы выходов приемников. Активный уровень “0” |
| 18 | 18 | R1IN | Вход приемника RS-232 |
| 19 | 19 | R1OUTB | Не инвертирующий выход приемника. Активен во всех режимах |
| 20 | 20 | R1OUT | Выход приемника RS-232 |
| 21 | 21 | R2OUT | Выход приемника RS-232 |
| 22 | 22 | R3OUT | Выход приемника RS-232 |
| 23 | 23 | R3IN | Вход приемника RS-232 |
| 24 | 24 | R2IN | Вход приемника RS-232 |
| 25 | 25 | C1- | Отрицательный вывод конденсатора для внутреннего импульсного преобразователя напряжения |
| 26 | 26 | UСС | Напряжение питания |
| 27 | 27 | C1+ | Положительный вывод конденсатора для внутреннего импульсного преобразователя напряжения |
| 28 | 28 | UH | +5,5 В вывод внутр. импульсного преоб. напр. |

Типовая схема подключения микросхемы представлена на рисунке 14, где

G1 – источник постоянного напряжения, UCC = (3,0…5,5) В;

С1…С5 – конденсаторы,

C1 = не менее 0,22 мкФ ± 20 %;

C5 = не менее 0,22 мкФ ± 20 %;

C2 = C3 = C4 = не менее 1,0 мкФ ± 20 %.

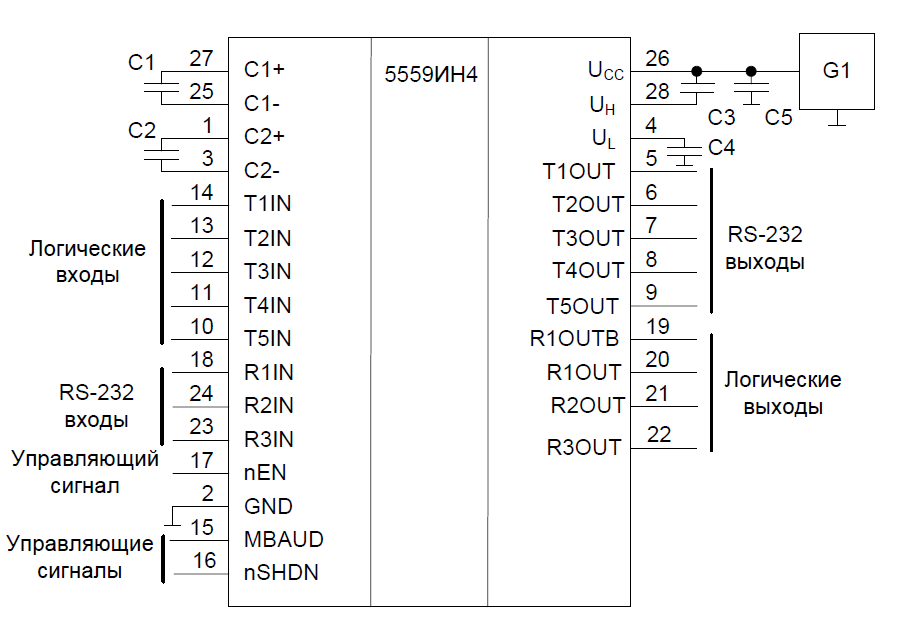


Рисунок 14 – Типовая схема подключения микросхемы приемопередатчика RS-232

## 1.4 Подключение отладочных плат

Для передачи данных между двумя отладочными платам необходимо соединить интерфейсные выводы между собой. Для интерфейсов CAN, USB, RS-232 есть специальные разъемы, в связи чем будут необходимы специальные кабели для их соединения. Интерфейс SPI будет подключен при помощи обычных проводов на разъеме общего назначения Х26. Схема соединений электрическая представлена в приложении В.

## 1.5 Алгоритмы работы программы

### 1.5.1 Алгоритм работы основной программы

Основная схема работы программы показана на рисунках 15 и 16.

После подачи питания на отладочную плату, происходит инициализация констант, настройка портов для LCD, настройка портов для кнопок, включение LCD и инициация таймера.



Рисунок 15 - Схема алгоритма основной программы



Рисунок 16 - Схема алгоритма основной программы (продолжение)

С помощью кнопок вверх («UP») и вниз («DOWN»), можно двигаться между строками в главном меню. В главном меню имеется 5 пунктов, которые отвечают за различные модули для тестирования: «CAN», «SSP», «USB», «UART», «FULL TEST». Выбор может быть осуществлен с помощью кнопки «SELECT». При выборе определенного пункта меню на экране появится информация о тесте и статусе прохождения теста (ожидание данных, данные переданы, данные получены, данные ошибочны, данные успешны). Выход из каждого теста в главное меню может быть осуществлен с помощью кнопки назад («LEFT»), при этом, если в ходе тестирования изменялись какие-либо настройки МК (например, как в тесте USB), то они вернутся в изначальное состояние.

### 1.5.2 Алгоритмы инициализации

На рисунках 17-19 показаны все алгоритмы инициализации, которые используются на всех этапах работы программы. Некоторые схемы алгоритма могут различаться для модулей Slave и Master, в связи с чем данные схемы алгоритма разделены на схемы «Master МК» и «Slave МК».



Рисунок 17 – Схемы алгоритмов инициации портов LCD, портов для кнопок, таймера, портов для CAN



Рисунок 18 – Схемы алгоритмов инициации портов SSP и запуска модулей SSP и UART



Рисунок 19 – Схемы алгоритмов инициации модуля CAN, запуска и отключения USB

### 1.5.3 Алгоритмы работы прерываний

На рисунках 20-22 представлены схемы работы прерываний в микроконтроллерах. Схемы алгоритмов, аналогично с предыдущим пунктом, разделены на «Master МК» и «Slave МК».

Прерывание по таймеру используется для обновления данных на LCD дисплее. В левом верхнем углу всегда находится фамилия исполнителя курсового проекта, в правом верхнем углу группа. Под группой находится информация о микроконтроллере («Master» или «Slave»). Затем следует 3 строчки главного меню или информации о проходящем в данный момент тесте. Внизу экрана находится бегущая строка, которая информирует пользователя, что данная система разработана в ходе выполнения курсовой работы по дисциплине «Микропроцессорные системы».



Рисунок 20 – Схема алгоритма прерывания таймера

Алгоритм обработки прерывания USB будет реализован только в Master, так как тестирование USB будет производиться только в этом микроконтроллере. Прерывание читает данные из буфера и обрабатывает их в зависимости от типа теста. Если в текущий момент не установлен флаг полного теста, то полученные данные будут сразу переданы обратно источнику. Если же флаг полного теста установлен, то полученные данные либо запишутся в память, либо будут переданы обработанные данные в зависимости от флага передачи данных.



Рисунок 21 – Схема алгоритма прерывания USB



Рисунок 22 – Схема алгоритма обработки прерывания CAN

Прерывание по CAN будет реализовано только в Slave МК, так как Master будет передавать данные и там нет необходимости в прерывании (хотя это возможно). После получения данные происходит проверка на флаг теста CAN. Если он установлен, то происходит проверка полученных данных. В результате проверки станет понятно, была ли передача успешна или данные ошибочны. Если флаг не установлен, то данные просто выводятся на экран.

### 1.5.4 Алгоритмы работы тестирующей программы CAN

На рисунке 23 представлена схема алгоритма тестирующей модуль CAN программы. Для Master и Slave данные программы различаются: они реализуют передачу и прием данных соответственно. Прием данных будет осуществляться по прерыванию, описанному в предыдущем пункте.



Рисунок 23 – Схема алгоритма теста CAN

### 1.5.5 Алгоритмы работы тестирующей программы SSP

Схема алгоритма программы, тестирующей модуль SSP представлена на рисунке 24. Master МК после инициализации модуля и настройки портов передает данные. В связи с этим Slave МК должен быть переведен в режим теста модуля SSP раньше, иначе данные приняты не будут и придется заново запускать данный режим на Master МК.



Рисунок 24 – Схема алгоритма теста модуля SSP

### 1.5.6 Алгоритмы работы тестирующей программы USB



Рисунок 25 – Схема алгоритма теста USB

На рисунке 25 представлена схема алгоритма для теста модуля USB. Данный модуль будет работать в режиме USB Device как CDC. Прием данных осуществляется по прерыванию и был описан в пункте 1.5.3. В связи с тем, что USB работает на более высокой частоте, чет МК, то приходится переводить МК на высокую частоту при запуске теста USB и при окончании переводить обратно.

### 1.5.7 Алгоритмы работы тестирующей программы UART

Тест UART будет производиться простейшей передачей данных с одного МК на другой сразу после инициации портов и модуля UART. Схема алгоритма данного теста представлена на рисунке 26.



Рисунок 26 – Схема алгоритма теста модуля UART

### 1.5.8 Алгоритмы работы полной тестирующей программы

Полная тестирующая программа представляет собой алгоритм, в ходе выполнения которого могут быть протестированы все имеющиеся в МК модули, отвечающие за интерфейсы связи двух МК.

Схема алгоритма полной тестирующей программы представлена на рисунках 28 – 30. По данному алгоритму данные попадают в Master МК из ПЭВМ по USB, затем они идут в Slave МК по SSP. Далее они возвращаются в Master МК по UART. После этого они должны быть переданы обратно в ПЭВМ по USB. В завершении, данные передаются в Slave МК по интерфейсу CAN. На этом полный тест завершен. Схематически данный алгоритм представлен на рисунке 27.



Рисунок 27 – Схема полного теста



Рисунок 28 – Схема алгоритма полного теста Master МК



Рисунок 29 – Схема алгоритма полного теста Master МК (продолжение)



Рисунок 30 – Схема алгоритма полного теста Slave МК

# 2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 2.1 Разработка программы

В ходе работы над курсовым проектом было задействовано следующее ПО: среда разработки Keil uVision 5 с дополнительными библиотеками «Standard Peripherals Library + software pack для Keil MDK 5» с официального сайта АО "ПКК Миландр". Внешний вид среды разработки представлен на рисунке 31.

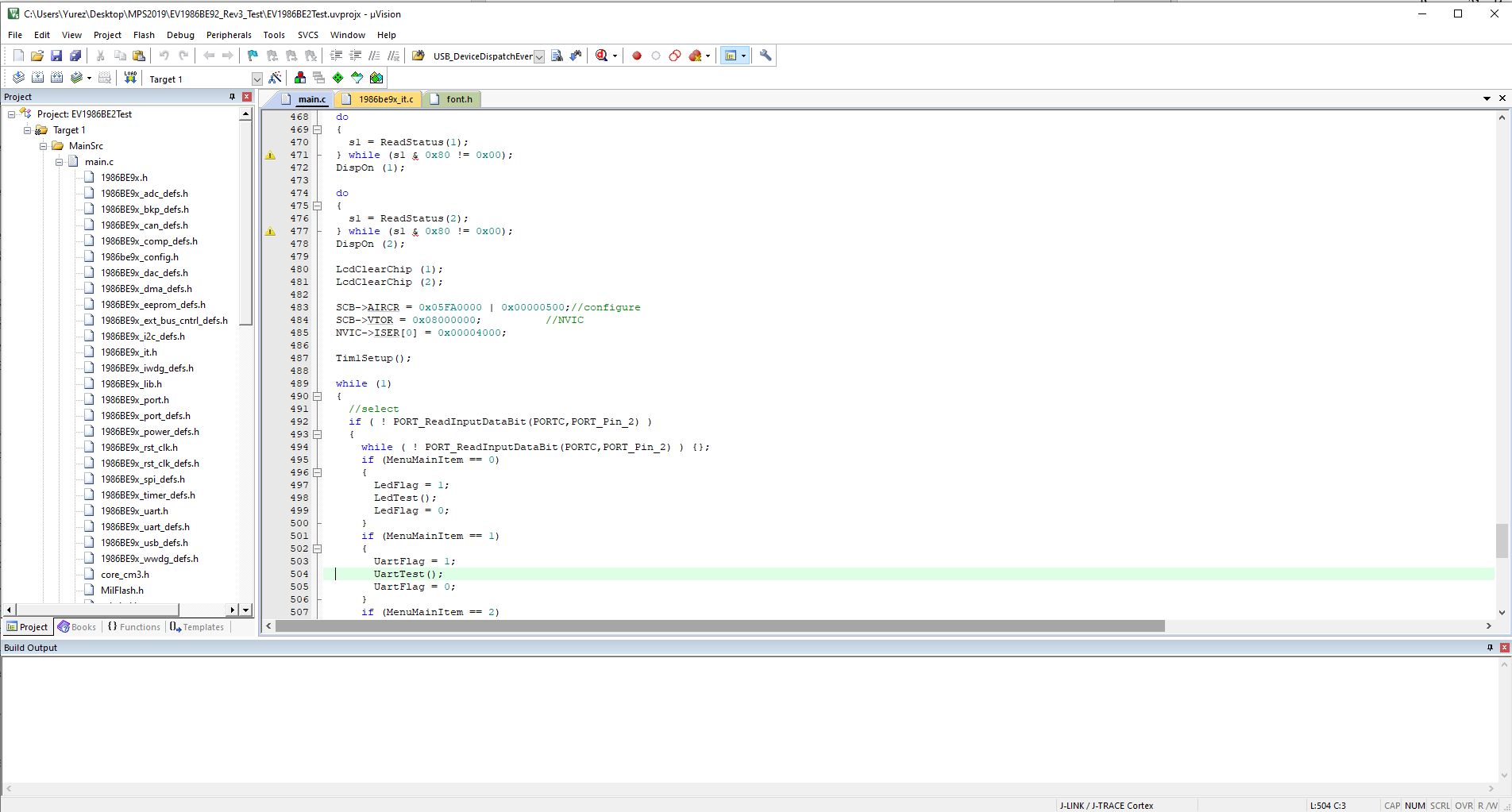


Рисунок 31 – Внешний вид Keil uVision 5

Среду можно использовать для разработки под самые разные платы, начиная от плат, основанных на МК архитектуры AVR и заканчивая архитектурой ARM. Кроме того, данная среда разработки может быть модифицирована с помощью библиотек, которые могут быть загружены с официальных сайтов разработчиков ИС и МК.

## 2.2 Программирование flash-памяти микроконтроллера, отладка программы

Программирование flash-памяти микроконтроллера производится с помощью JTAG/SW программатора. При работе в режиме отладки разрешается работа отладочного интерфейса JTAG/SW. При этом к микроконтроллеру может быть подключен JTAG/SW адаптер, с помощью которого программные средства разработки позволяют работать с микроконтроллером в отладочном режиме. Линии JTAG должны быть подтянуты к питанию сопротивлениями не менее 10К с учетом, чтобы эти подтяжки не влияли на работу системы. Схема подключения при отладке представлена на рисунке 32.

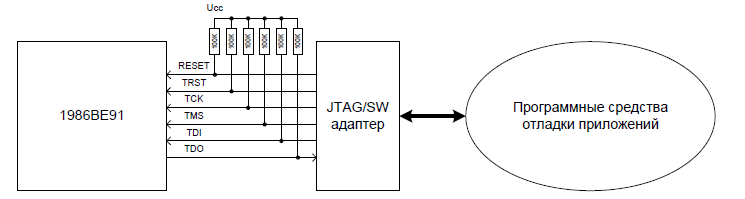


Рисунок 32 – Схема работы в режиме отладки

В отладочном режиме можно:

* стирать, записывать, считывать внутреннюю Flash-память программ;
* считывать и записывать содержимое ОЗУ, периферии;
* выполнять программу в пошаговом режиме;
* запускать программу в нормальном режиме;
* останавливать программу по точкам остановки;
* просматривать переменные выполняемой программы;
* проводить трассировку хода выполнения программного обеспечения.

Hex файл для МК «Master» занимает 41Кб памяти, для МК «Slave» занимает 30Кб памяти. Это связано с тем, что второй МК не содержит библиотек и модулей для работы с USB, так как его тестирование производится только на МК «Master».

По данным, полученным с помощью точки останова, подготовка к запуску занимает 0,2 секунды до момента возможности выбора тестирования.

Оценка скорости работы модулей, отвечающих за работу с интерфейсами, не представляется возможным в связи с тем, что невозможно поставить точки останова в программах обоих микроконтроллеров, а с визуальной точки зрения передача происходит мгновенно.

## 2.3 Программирование USB драйвера

В связи с отсутствием необходимого оборудования для тестирования режима USB Host на отладочной плате, было решено протестировать режим USB Device. Для упрощения работы с компьютером было решено использовать МК в качестве CDC устройства (USB communications device class— является составным классом устройства универсальной последовательной шины). Для работы с USB на компьютере использовался драйвер, после установки которого МК распознаваться, как виртуальный COM порт (рисунок 33).

Исходный код драйвера представлен в приложении Б аналогично исходным кодами обоих МК.

Для работы с виртуальным COM портом была установлена программа COM PORT Tollkit, которая позволяет принимать и отправлять сообщения. Интерфейс с примером работы данной программы представлен на рисунке 34.

Кроме того, по данным с рисунка 34 можно оценить задержки при передаче данных по каналу USB с компьютера к МК и обратно. Данная задержка в среднем составляет примерно 0,02 секунды.

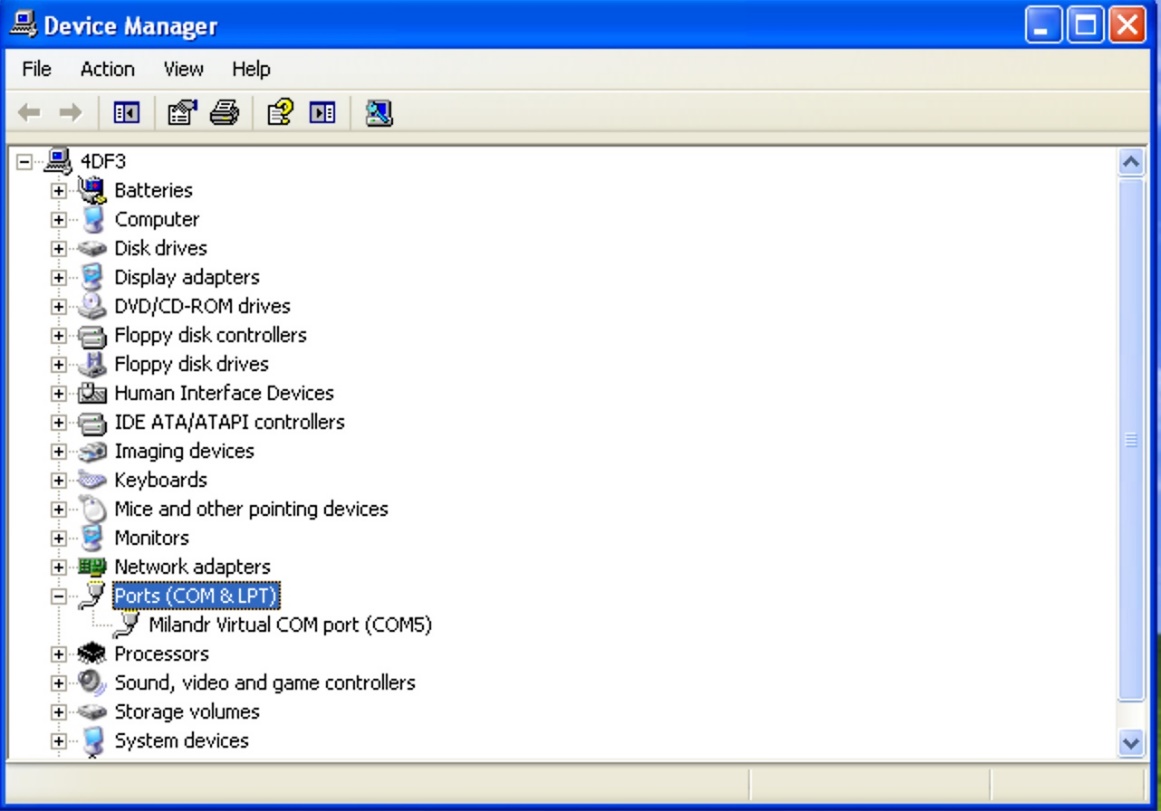


Рисунок 33 – МК в виде виртуального COM порта

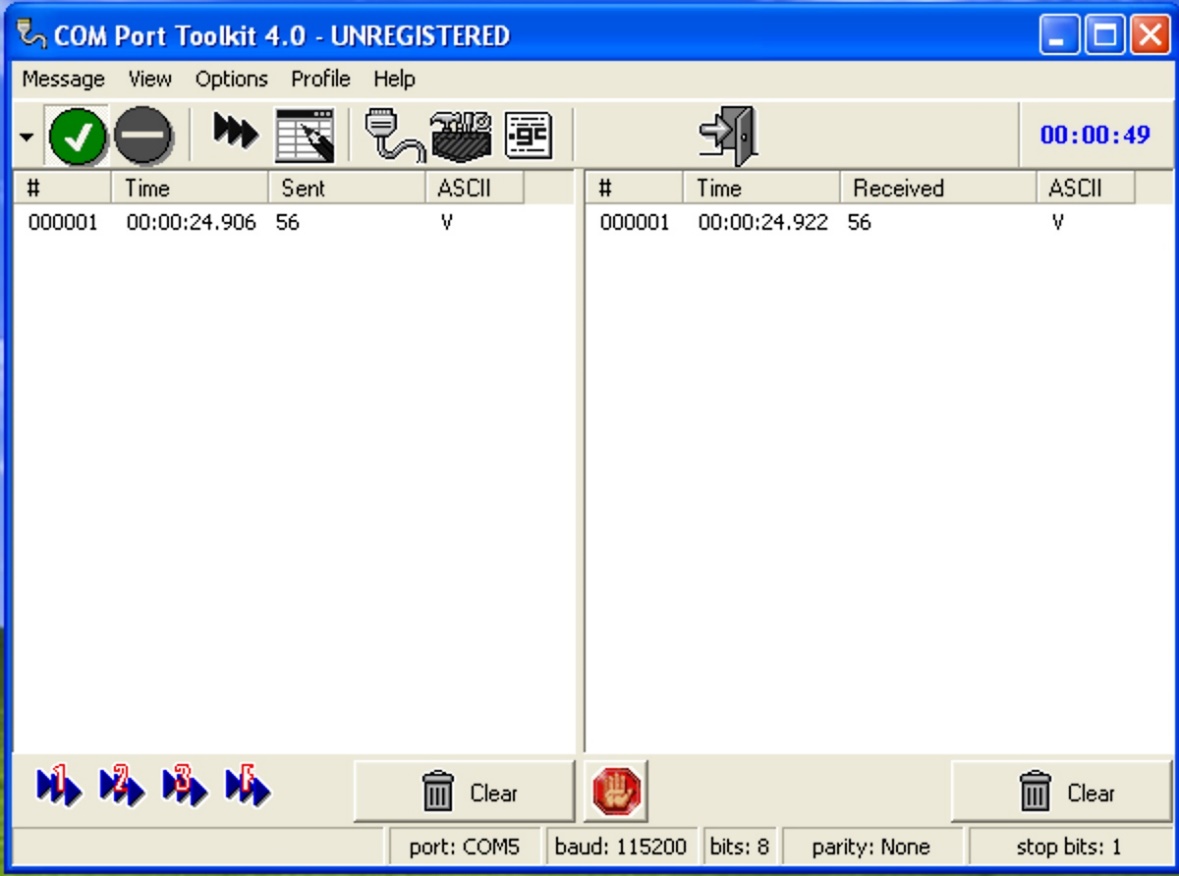


Рисунок 34 - COM PORT Tollkit

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта был изучен МК К1986ВЕ92QI, отладочная плата для него и интерфейсы взаимодействия двух отладочных плат. Были написаны тестирующие программы для каждого модуля по-отдельности и всех модулей в ходе одного теста.

На первом этапе работ, были изучены интерфейсы USB, CAN, а также модули, отвечающие за работу интерфейсов UART, SPI, USB, CAN в МК. В ходе работы была получена функциональная схема устройства, показывающая, какие модули присутствуют в микроконтроллере, какие из них используются и как различные модули двух МК взаимодействуют между собой. В ходе разработки принципиальной схемы было получено представление о модулях, которые используются для работы с различными интерфейсами. Схема соединений содержит информацию о том, как и чем соединять две отладочные платы между собой для корректной их работы.

Для разработки использовалась интегрированная среда разработки Keil uVision 5 и программатор JTAG/SW для отладки полученной программы.

Для работы МК с ПЭВМ использовался драйвер, который позволяется распознать в микроконтроллере USB устройство и использовать его как CDC.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация на микроконтроллер К1986ВЕ92QI [Электронный ресурс]. – URL: https://ic.milandr.ru/products/mikrokontrollery\_i\_protsessory/32\_razryadnye\_mikrokontrollery/1986ve9kh\_yadro\_arm\_cortex\_m3/k1986ve92qi/#main\_tab (дата обращения 25.11.2019).
2. Официальный сайт Миландр [Электронный ресурс]. – URL: https://ic.milandr.ru/ (дата обращения 25.11.2019).
3. ПРОГРАММИРОВАНИЕМИКРОКОНТРОЛЛЕРА К1986ВЕ92QI КОМПАНИИ «МИЛАНДР» [Электронный ресурс]. – URL: https://edu.milandr.ru/upload/iblock/8bd/8bda469e07e2dd755e1880d3543bf613.pdf (дата обращения 25.11.2019).
4. Форум Миландр [Электронный ресурс]. – URL: http://forum.milandr.ru (дата обращения 25.11.2019).
5. Графический ЖК-модуль MT-12864J фирмы МЭЛТ [Электронный ресурс]. – URL: http://www.gaw.ru/html.cgi/txt/lcd/lcm/melt/graf/MT-12864J.htm (дата обращения 25.11.2019).
6. Start Milandr [Электронный ресурс]. – URL: https://startmilandr.ru/doku.php/start (дата обращения 25.11.2019).
7. Высокоскоростные приемопередатчики CAN - шины [Электронный ресурс]. – URL: http://www.gaw.ru/html.cgi/txt/ic/Atmel/standart\_app/automotive/ATA6660.htm (дата обращения 25.11.2019).
8. Микросхема приемопередатчика интерфейса RS-232 [Электронный ресурс]. – URL: https://ic.milandr.ru/products/interfeysnye\_mikroskhemy/rs232/5559in4u/ (дата обращения 25.11.2019).
9. ГОСТ 2.743-91. Единая система конструкторской документации. Обозначения условные графические в схемах. Элементы цифровой техники.
10. ГОСТ 2.701-2008. Единая система конструкторской документации. Схемы. Виды и типы. Общие требования к выполнению.
11. ГОСТ 2.764-86. Обозначения условные графические в электрических схемах. Интегральные оптоэлектронные элементы индикации.
12. ГОСТ 2.755-87. Обозначения условные графические в электрических схемах. Устройства коммутационные и контактные соединения, Москва, 1969 г., - 41 с.