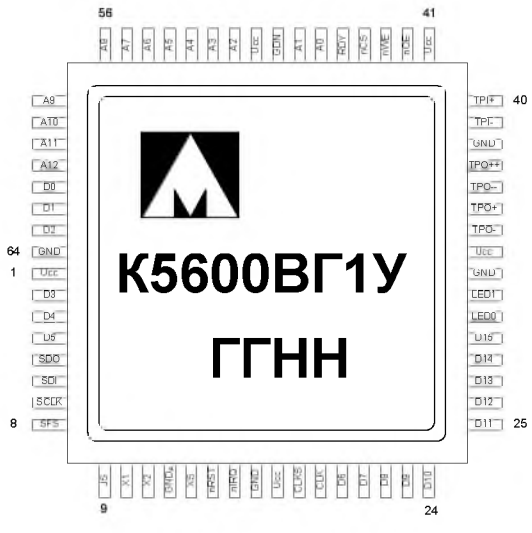
Микросхема контроллера ЛВС уровня звена данных протокола IEEE 802.3/Ethernet 5600ВГ1У, К5600ВГ1У, К5600ВГ1УК



ГГ – год выпуска
НН – неделя выпуска

Основные характеристики микросхемы:

- Входное напряжение от 4,5 В до 5,5 В;
- Статический ток потребления I_{CCS} не более 20 мА;
- Статический ток потребления (приемопередатчики в состоянии выключено) I_{CCS1} не более 50 мкА;
- Динамический ток потребления не более 190 мА;
- Ток короткого замыкания по выходам «TDOp» и «TDOn» 50 ÷ 100 мА;
- Время выхода в рабочий режим не более 200 мкс;
- Рабочий диапазон температур:

Обозначение	Диапазон
5600ВГ1У	минус 60 – 85 °С
К5600ВГ1У	минус 60 – 85 °С
К5600ВГ1УК	0 – 70 °С

Тип корпуса:

– 64-х выводной металлокерамический корпус Н18.64-1В.

Общее описание и области применения микросхемы

Контроллер ЛВС предназначен для использования в устройствах локальной вычислительной сети на основе протоколов 10BASE-T IEEE802.3/Ethernet с применением специализированных вычислительных устройств и ПК и работы под контролем управляющего устройства.

Контроллер реализует прием и передачу данных с контролем ошибок.

Содержание

1 Описание выводов

Таблица 1 – Описание выводов микросхемы

№ вывода корпуса	Функциональное назначение выводов	Обозначение
1	Питание	U_{CC}
2	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D3
3	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D4
4	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D5
5	Цифровой выход данных последовательного порта. В тестовом режиме сигнал TDO порта JTAG	SDO/TDO
6	Цифровой вход данных последовательного порта. В тестовом режиме сигнал TDI порта JTAG	SDI/TDI
7	Цифровой вход. Сигнал тактовой частоты обмена по последовательному порту. В тестовом режиме сигнал TCK порта JTAG	SCLK/TCK
8	Цифровой вход. Сигнал стробирования обмена по последовательному порту. В тестовом режиме сигнал TMS порта JTAG	SFS/TMS
9	Цифровой вход. Сигнал переключения в тестовый режим. Также сигнал nTRST порта JTAG.	JS/nTRST
10	Аналоговый вход кварцевого резонатора	X1
11	Аналоговый выход кварцевого резонатора	X2
12	Общий (аналоговый)	GND_A
13	Выбор режима работы от внешнего генератора или синтезатора частоты	XS
14	Цифровой вход. Сброс. Активный низкий уровень	nRST
15	Цифровой выход. Запрос прерывания. Активный низкий уровень.	nIRQ
16	Общий	GND
17	Питание	U_{CC}
18	Цифровой вход. Сигнал выбора режима записи через параллельный порт (0 – по положительному фронту сигнала nWE; 1 – по отрицательному фронту сигнала nWE)	CLKS
19	Цифровой вход. Сигнал рабочей частоты (управляет только снятием 3-го состояния с выхода данных в цикле чтения и формированием сигнала RDY)	CLK
20	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D6
21	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D7
22	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D8
23	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D9
24	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D10

№ вывода корпуса	Функциональное назначение выводов	Обозначение
25	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D11
26	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D12
27	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D13
28	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D14
29	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D15
30	Цифровой выход индикации коллизии/режима работы	LED0
31	Цифровой выход индикации состояния соединения	LED1
32	Общий	GND
33	Питание	U_{CC}
34	Аналоговый выход линейного интерфейса, отрицательный	TPO-
35	Аналоговый выход линейного интерфейса, положительный	TPO+
36	Аналоговый выход линейного интерфейса, отрицательный задержанный	TPO- -
37	Аналоговый выход линейного интерфейса, положительный задержанный	TPO++
38	Общий	GND
39	Аналоговый вход линейного интерфейса, отрицательный	TPI-
40	Аналоговый вход линейного интерфейса, положительный	TPI+
41	Питание	U_{CC}
42	Цифровой вход. Сигнал строба цикла обмена. Разрешает прием/передачу данных. Активный низкий уровень	nOE
43	Цифровой вход. Сигнал записи. Управляет направлением передачи данных по шине данных. Активный низкий уровень	nWE
44	Цифровой вход. Сигнал выборки кристалла. Активный низкий уровень	nCS
45	Цифровой выход. Сигнал готовности. Активный высокий уровень. Выставляется по положительному фронту сигнала CLK	RDY
46	Цифровой вход шины адреса	A0
47	Цифровой вход шины адреса	A1
48	Общий	GND
49	Питание	U_{CC}
50	Цифровой вход шины адреса	A2
51	Цифровой вход шины адреса	A3
52	Цифровой вход шины адреса	A4
53	Цифровой вход шины адреса	A5
54	Цифровой вход шины адреса	A6
55	Цифровой вход шины адреса	A7
56	Цифровой вход шины адреса	A8
57	Цифровой вход шины адреса	A9
58	Цифровой вход шины адреса	A10
59	Цифровой вход шины адреса	A11
60	Цифровой вход шины адреса	A12

№ вывода корпуса	Функциональное назначение выводов	Обозначение
61	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D0
62	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D1
63	Цифровой вход/выход двунаправленной шины данных с z-состоянием. Управляется сигналами nOE и nWE	D2
64	Общий	GND

2 Условно-графическое обозначение

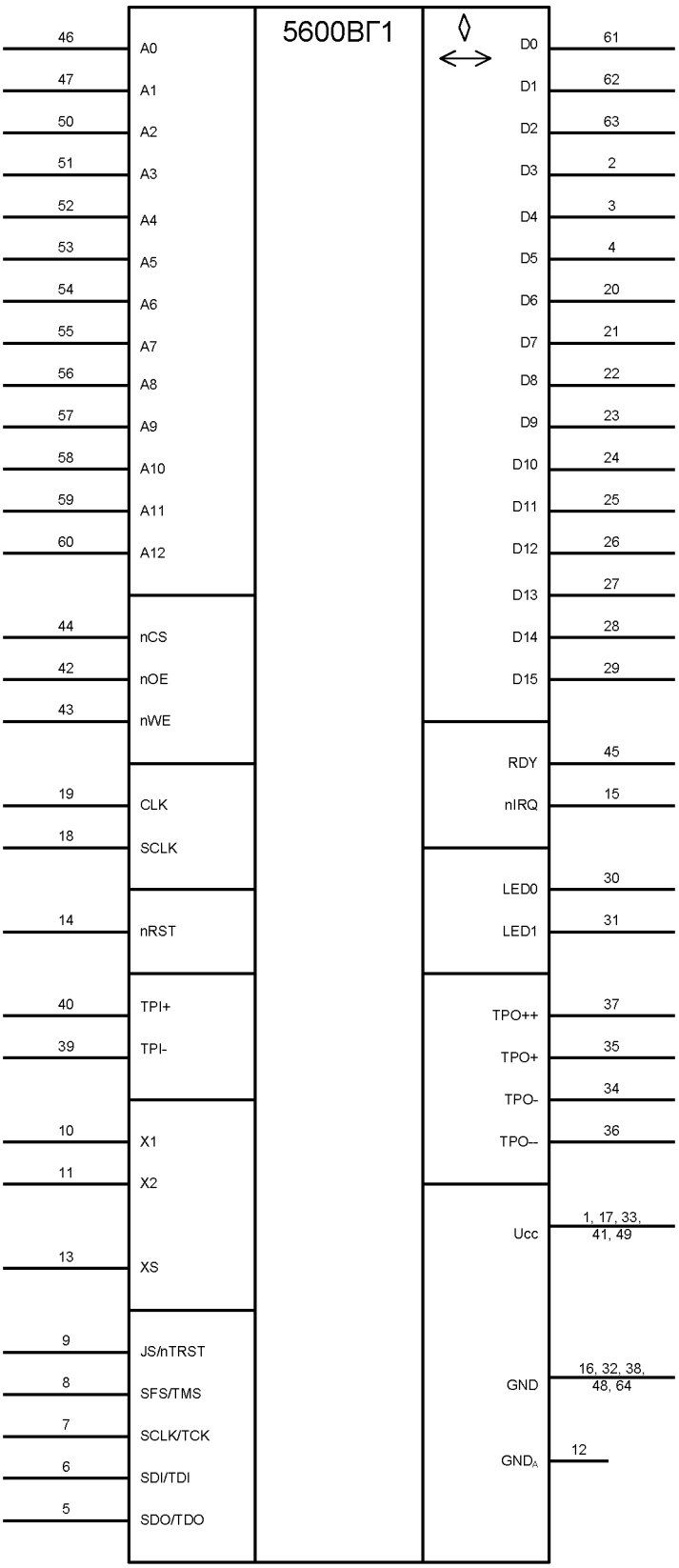


Рисунок 1 – Условно-графическое обозначение

3 Структурная блок-схема микросхемы

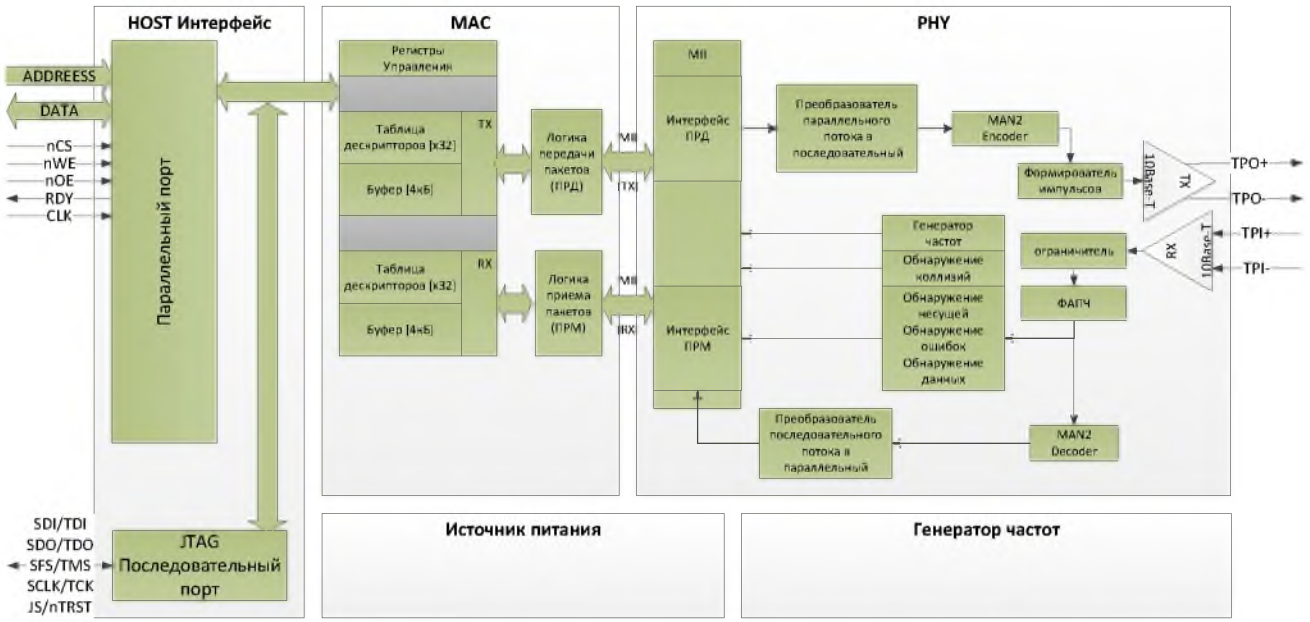


Рисунок 2 – Структурная блок-схема

4 Общее описание микросхемы

4.1 Технические характеристики

Контроллер ЛВС 5600ВГ1У (далее контроллер ЛВС) имеет один двунаправленный канал 10BASE-T IEEE 802.3/Ethernet.

В контроллере ЛВС реализованы:

- физический (PHY) уровень протокола 10BASE-T IEEE 802.3/Ethernet;
- уровень звена данных (MAC) протокола Ethernet 10BASE-T IEEE 802.3, в составе из буферов, таблиц дескрипторов, приемника и передатчика (ПРМ и ПРД соответственно);
- буферы принимаемых и передаваемых сообщений общей емкостью 8 кБ;
- параллельный интерфейс сопряжения контроллера с управляющим устройством – 16-и разрядный интерфейс с раздельными шинами адреса и данных;
- последовательный интерфейс сопряжения контроллера с управляющим устройством – 4-проводный дуплексный с отдельными выводами тактового и кадрового синхросигналов;
- настройка параметров соединений в неавтоматическом режиме;
- индикация состояний соединения.

В комплект поставки микросхемы входит драйвер, предназначенный для обеспечения взаимодействия встроенного ПО управляющего устройства (далее УУ) с контроллером TMS320C50/C54x. Он предоставляет только функции и процедуры для доступа к регистрам управления, к регистрам состояния контроллера и к буферам данных.

Контроллер ЛВС предназначен для работы в среде, в которой:

- объединение узлов в ЛВС производится посредством коммутатора типа HUB на основе 5600ВВ2У;
- средой передачи сигнала в ЛВС является витая пара;
- настройка параметров соединений производится в неавтоматическом режиме.

4.2 Указания по применению и эксплуатации

При ремонте аппаратуры и измерении параметров микросхем замену микросхем необходимо проводить только при отключенных источниках питания.

Инструмент для пайки (сварки) и монтажа не должен иметь потенциал, превышающий 0,3 В относительно шины «Общий».

Запрещается подведение каких-либо электрических сигналов (в том числе шин «Питание», «Общий») к выводам микросхем, не используемым согласно схеме электрической.

Если не используется Тестовый порт вывод 9 (JS_nTRST) необходимо подключить к шине «Общий».

Если не используется Последовательный порт вывод 7 (SCLK/TCK) необходимо подключить к шине «Общий».

Остальные выводы Последовательного/тестового порта (5, 6, 8) рекомендуется подключить к шине «Общий».

4.3 Регистры управления

Регистры контроллера ЛВС и их начальные значения описаны в таблице 2

Таблица 2 – Регистры управления контроллера

Наименование	Номер регистра	Значение по умолчанию	Описание
MAC_CTRL	0x00	0xC0A0	Регистр управления MAC уровнем контроллера
MinFrame	0x01	0x0040	Регистр задания минимальной длины пакета
MaxFrame	0x02	0x0600	Регистр задания максимальной длины пакета
CollConfig	0x03	0x0000	Регистр управления обработкой коллизий
IPGTx	0x04	0x0006	Регистр задания межпакетного интервала (в тактах PHY = 100ns)
MAC_ADDR_T	0x05	0x89AB	Регистр младшей части MAC-адреса
MAC_ADDR_M	0x06	0x4567	Регистр средней части MAC-адреса
MAC_ADDR_H	0x07	0x0123	Регистр старшей части MAC-адреса
HASH0	0x08	0x0000	Регистр слова 0 (младшего слова) HASH таблицы
HASH1	0x09	0x0000	Регистр слова 1 HASH таблицы
HASH2	0x0A	0x0000	Регистр слова 2 HASH таблицы
HASH3	0x0B	0x8000	Регистр слова 3 (старшего слова)HASH таблицы
INT_MSK	0x0C	0x0000	Регистр маски прерываний
INT_SRC	0x0D	0x0000	Регистр флагов прерываний
PHY_CTRL	0x0E	0x81D0	регистр управления PHY уровнем контроллера
PHY_STAT	0x0F	0xXXXX	Регистр состояния PHY уровня
RXBF_HEAD	0x10	0x07FF	Регистр старшей части буфера приемника (управляется программистом и указывает на последний свободный адрес в буфере)
RXBF_TAIL	0x11	0x0000	Регистр младшей части буфера приемника (управляется контроллером и указывает на первый свободный адрес в буфере)
STAT_RX_ALL	0x14	0x0000	Счетчик количества входящих пакетов дошедших до MAC-уровня
STAT_RX_OK	0x15	0x0000	Счетчик количества успешно принятых входящих пакетов (при выставленном бите ERROR_FRAME_EN – считает все пакеты)
STAT_RX_OVF	0x16	0x0000	Счетчик количества входящих пакетов, вызвавших переполнение буфера ПРМ
STAT_RX_LOST	0x17	0x0000	Счетчик количества входящих пакетов, потерянных из-за неготовности MAC-уровня к приему (неготовность дескриптора)
STAT_TX_ALL	0x18	0x0000	Счетчик количества исходящих пакетов
STAT_TX_OK	0x19	0x0000	Счетчик количества успешно отосланных исходящих пакетов
	0x12..0x1E		Зарезервировано
GCTRL	0x1F	0x4382	Регистр управления стыка с HOST

Примечание – Подробное описание битов регистров приведено в Приложении 1.

4.4 Внешние интерфейсы

Контроллер ЛВС имеет два интерфейса сопряжения с другими устройствами: параллельную шину с раздельными адресом и данными и двунаправленный последовательный порт типа SPI.

Параллельный порт представлен 13-разрядной шиной адреса и 16-разрядной шиной данных и является аналогом интерфейса TMS320 C5x/C54x.

Последовательный порт типа SPI имеет общие сигналы строба и частоты ПРМ и ПРД контроллера ЛВС. Обмен по последовательному порту осуществляется в пакетном режиме. Пакет состоит в общем случае из байта команды и её параметров. Для команд чтения и записи пакеты состоят из соответствующей команды, сопровождаемой начальным адресом и последовательностью слов данных. Для каждого последующего слова данных адрес чтения/записи инкрементируется в пределах адресного пространства контроллера. Предельная частота обмена по SPI – 20 МГц.

По сбросу контроллер ЛВС устанавливается в режим работы через параллельный порт.

Переключение между портами возможно в процессе работы контроллера только через последовательный порт. В режиме работы через последовательный порт работа через параллельный порт не доступна.

Подробное описание приведено в разделе «Интерфейс внешней шины».

4.5 Адресное пространство

В контроллере ЛВС имеется встроенные независимые циклические буферы ПРМ и ПРД емкостью 2К 16-разрядных слов каждый.

Также для управления параметрами передачи в контроллере ЛВС имеются таблицы дескрипторов пакетов ПРМ и ПРД. 32 дескриптора пакетов ПРМ и 32 дескриптора пакетов ПРД (по 8 байт каждый).

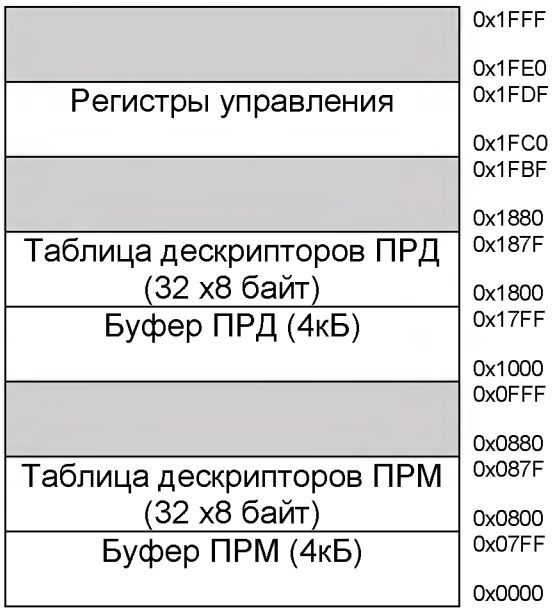


Рисунок 3 – Карта адресного пространства 5600ВГ1У

Примечание:

Адресация в 16-разрядных словах.

Чтение/запись в буферы и таблицы со стороны управляющего устройства возможна одновременно с приёмом и/или передачей данных.

4.6 Дескрипторы

Дескрипторы предназначены для задания параметров приёма/передачи отдельных пакетов и отражения состояния приема и передачи соответственно. Для приема и передачи имеется 2 таблицы по 32 записи/дескриптора каждая. Порядок и назначение полей дескрипторов приведены в таблице 2.

Таблица 2 – Назначение полей дескрипторов

Обозначение	Смещение	Назначение
CTRL	0	поле управления и состояния пакета
LEN	1	длина пакета
Reserved	2	зарезервировано
ADDR	3	адрес начала блока данных пакета

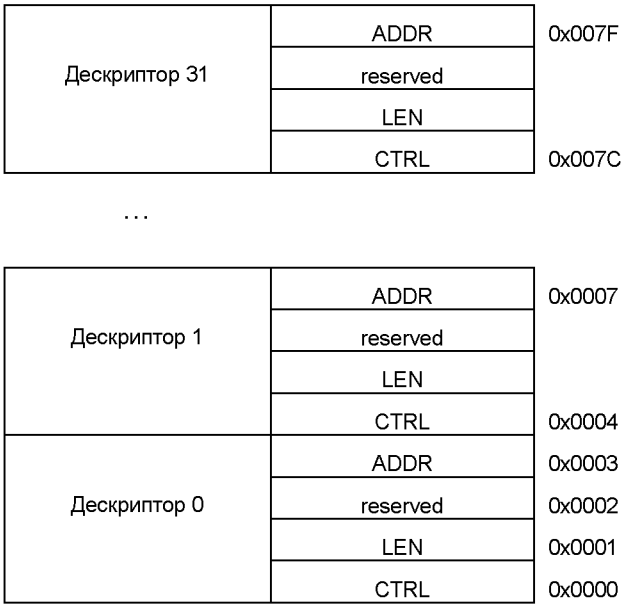


Рисунок 4 – Порядок расположения дескрипторов в адресном пространстве

Примечание – Подробное описание полей управления дескрипторов ПРД и ПРМ приведено в Приложении 2 и Приложении 3 соответственно.

4.7 Ядро MAC IEEE 802.3/Ethernet

Ядро MAC контроллера ЛВС соответствует стандарту IEEE 802.3/Ethernet, и поддерживает работу в дуплексном, так и в полудуплексном режимах, включая обнаружение ошибок, обнаружение и вставку преамбулы пакета, а также вычисление, вставку и проверку контрольной суммы CRC.

Регистры управления позволяют производить гибкую настройку режима работы, управлять параметрами приёма/передачи, дополнять пакеты до минимальной длины (по стандарту IEEE 802.3).

4.8 Физический уровень (PHY)

В контроллер ЛВС встроен модуль физического уровня (PHY) протокола IEEE802.3/Ethernet. Данный модуль включает кодер и декодер кода Манчестер-2, ФАПЧ, автоматический фазовый детектор.

Контроллер ЛВС производит обнаружение наличия подключения к линии и формирование сигналов присутствия на линии и имеет аналоговые приёмопередающие каскады для упрощённого подключения к линии при помощи трансформаторных развязок.

Данный модуль предназначен для работы по стандарту 10BASE-T и обеспечивает скорость подключения до 10 Мбит/с.

4.9 Драйвер

Программное обеспечение (драйвер) предназначено для предоставления программисту средств управления контроллером. Драйвер включает в себя макроопределения регистровой модели устройства, определение регистровых структур и базовые функции для приёма, отправки пакетов и проверки контроля результата их отправки. Подробное описание приведено в Приложении 5.

5 Описание функционирования микросхемы

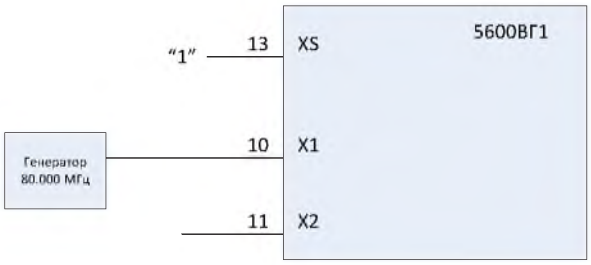
Контроллер ЛВС выполняет две основные функции — прием и передачу пакетов Ethernet.

Для правильной работы контроллера ЛВС необходима предварительная настройка.

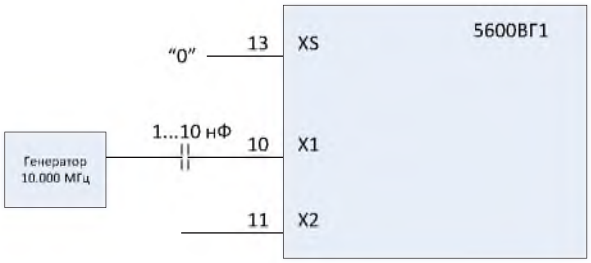
5.1 Тактирование

Основная рабочая частота контроллера 80 МГц. Контроллер содержит PLL с опорным синхросигналом 10 МГц от резонатора (подключение к выводам X1 и X2) или 80 МГц от генератора (режим обхода, подключение в выводу X1). Режим работы определяется выводом XS («1» в режиме обхода):

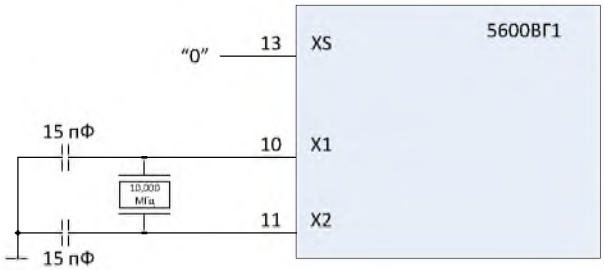
- Для перевода микросхемы *в режим обхода* на вывод XS подается высокий уровень, при этом на вход X1 подается тактовая частота 80 МГц.
- Для перевода микросхемы *в режим умножения* на вывод XS подается низкий уровень. При этом микросхема должна получать опорную частоту 10 МГц. В качестве источника опорной частоты может выступать кварцевый резонатор (на выводах X1 и X2) или генератор (см. Рисунок 5). В случае применения 10 МГц генератора, он должен быть подключен к входу X1 через емкость 1…10 нФ.



Режим обхода



Режим умножения с внешним генератором



Режим умножения с внешним кварцевым резонатором

Рисунок 5 – Схемы включения внешних источников тактирования

5.2 Интерфейс внешней шины

Интерфейс внешней шины предназначен для связи контроллера ЛВС с внешним устройством управления (далее по тексту УУ), являющегося также источником данных. Интерфейс внешней шины состоит из двух компонентов: параллельного и последовательного портов.

В процессе работы контроллера возможно переключение на работу по последовательному порту и обратно подачей соответствующей команды управления через последовательный порт.

По сбросу контроллер ЛВС устанавливается в режим работы через параллельный порт.

5.2.1 Параллельный порт

Параллельный порт является основным средством взаимодействия с УУ. Он представлен раздельными 13-разрядной шиной адреса, 16-разрядной шиной данных и шиной управления. Параллельный порт имеет интерфейс доступа к памяти из 3-х сигналов:

- выборки устройства (nCS);
- строба чтения (nOE);
- сигнала записи (nWE).

Контроллер формирует сигнал RDY для обмена с асинхронными устройствами.

В контроллере 5600ВГ1У имеется два основных режима работы, различающихся способом выставления сигнала RDY, управляемых через внешний контакт CLKS или битом ASYNC_MODE регистра GCTRL. В обоих режимах данные захватываются по завершении цикла обмена (нарастающему фронту сигнала nWE). Чтение начинается после выставления сигнала nOE при активном уровне сигнала nCS и неактивном уровне сигнала nWE. Сигнал RDY информирует о готовности данных при чтении или готовности принять данные при записи (внутренний цикл записи инициируется после завершения цикла – снятие управляющих сигналов).

В режиме 1 (CLKS = 0 и ASYNC_MODE = 0) сигнал RDY выставляется синхронно по внешнему сигналу CLK.

В режиме 2 (CLKS = 1 или ASYNC_MODE = 1) сигнал RDY выставляется по внутреннему тактовому сигналу асинхронно внешнему сигналу CLK.

Таблица 3 содержит перечень, тип и назначение выводов контроллера ЛВС, обеспечивающих подключение к внешней шине.

Таблица 3 – Назначение выводов интерфейса параллельного порта

Наименование	Направление	Описание
CLK	I	Вход тактовой частоты
A[12 – 0]	I	13-разрядная параллельная шина адреса
D[15 – 0]	I/O	16-разрядная параллельная шина данных
nOE	I	Разрешение чтения 0 – чтение 1 – неактивное состояние выводов D
nWE	I	Разрешение записи 0 – запись 1 – чтение
nCS	I	Сигнал выборки ИМС 0 – выбрана 1 – не выбрана, неактивное состояние выводов D
RDY	O	Сигнал готовности 0 – цикл не завершен 1 – цикл завершен
nIRQ	O	Сигнал запроса прерывания (активный уровень «0»)
nRST	I	Сигнал сброса (активный уровень «0»)

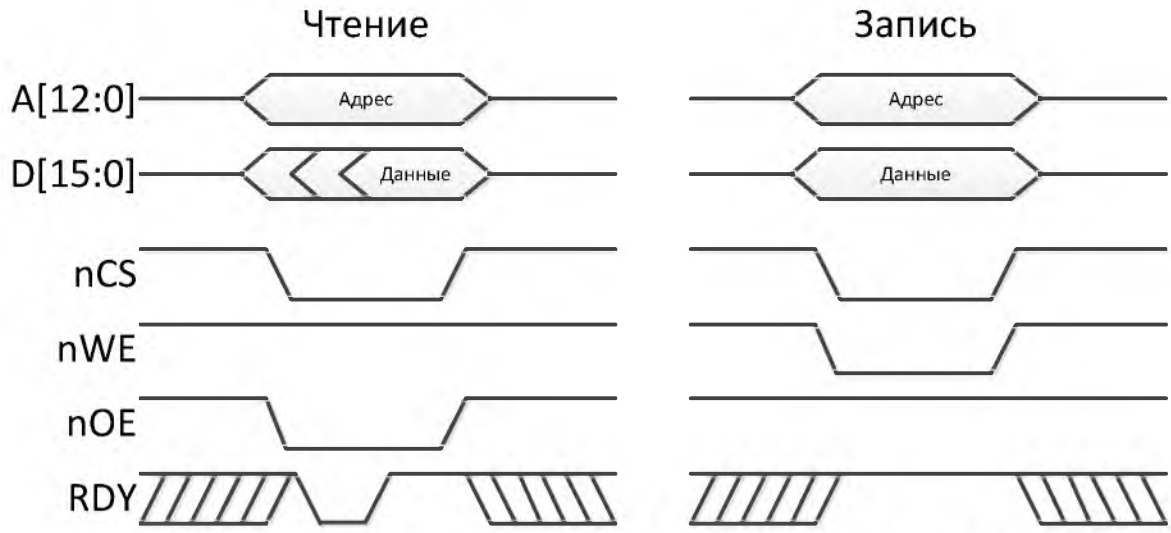


Рисунок 6 – Диаграммы чтения/записи параллельного порта

Примечания:

1. Не допускаются блочные транзакции (без снятия сигналов nCS, nWE и nOE/nRE).
2. Рекомендуется использовать длительности цикла обмена не менее 125 нс.
3. Не рекомендуется использовать частоту работы внешней шины выше 40 МГц.

5.2.2 Последовательный порт

Последовательный порт является вспомогательным и предназначен для использования в схемах с малоинтенсивным обменом и с ограниченными возможностями по подключению с УУ.

Таблица 4 содержит перечень, тип и назначение выводов контроллера ЛВС, обеспечивающих подключение к внешней шине.

Таблица 4 – Назначение выводов интерфейса последовательного порта

Наименование	Направление	Описание
SFS	I	Строб кадра ПРД
SDO	O	Выход данных ПРД
SCLK	I	Вход частоты ПРД
SDI	I	Вход данных ПРМ

Неактивный уровень строба данных («1») приводит к прерыванию чтения/записи.

По фронту сигнала SCLK данные меняются, по срезу сигнала SCLK данные фиксируются.

Обращение через последовательный порт осуществляется в пакетном режиме. Пакет (кадр) представляет собой последовательность байт и его размер определяется сигналом SFS. Все время передачи кадра сигнал SFS должен удерживаться в «0». Между передачей кадров сигнал SFS устанавливаться в «1». Размер пакета определяется числом полных переданных байт в период низкого уровня SFS.

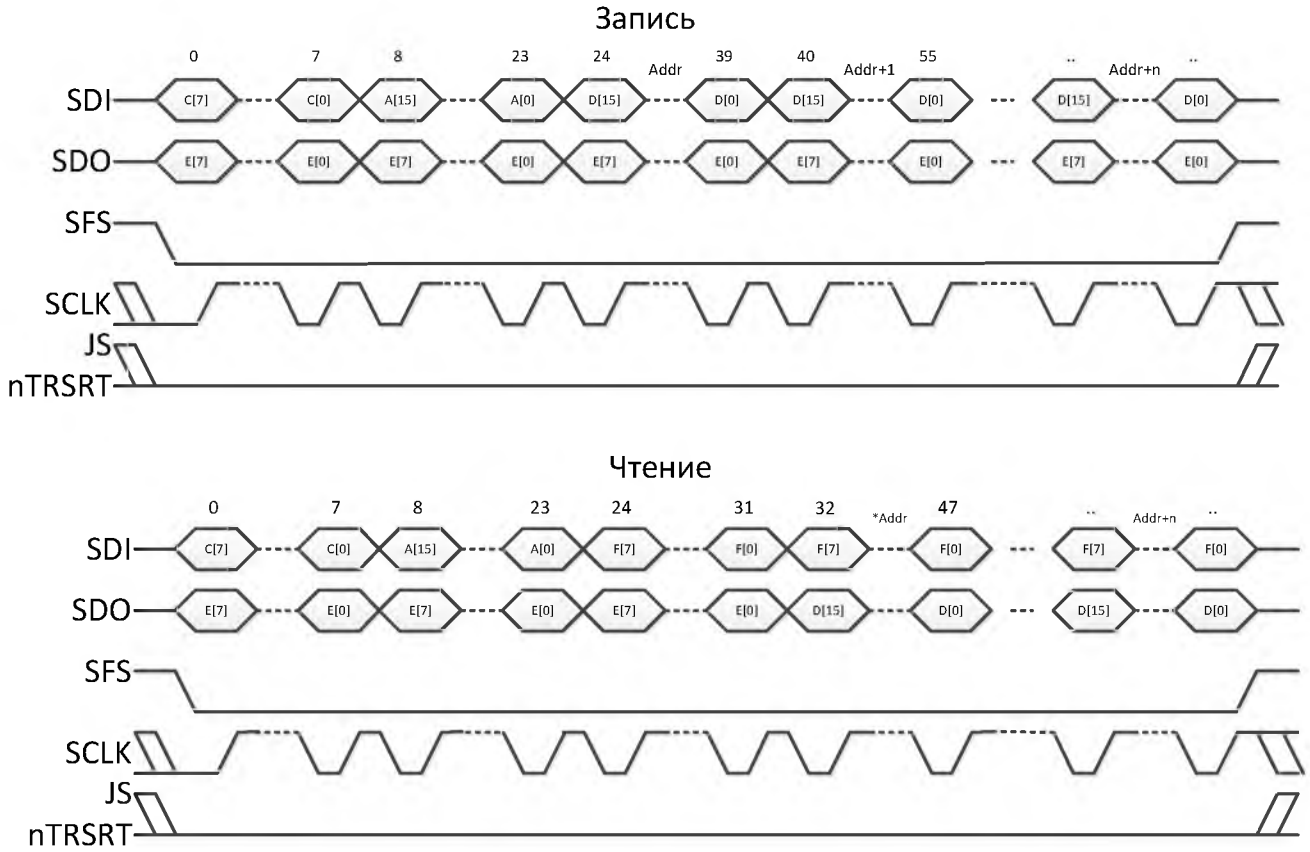


Рисунок 7 – Диаграммы чтения/записи последовательного порта

C[7..0] – код команды;
A[15..0] – начальный адрес (для команд записи/чтения);
D[15..0] – данные (для команд записи/чтения);
F[7..0] – заполнитель на время приёма данных (0x00);
E[7..0] – код ошибки

Таблица 5 – Описание команд

Наименование	Значение	Описание
Cmd_RESET	0xFF	Сброс устройства
Cmd_selectSPI	0x18	Режим работы через последовательный порт
Cmd_selectPPI	0x24	Режим работы через параллельный порт
Cmd_Write	0x3C	Запись блока данных
Cmd_Read	0x42	Чтение блока данных
Коды ошибок		
NoErr	0x00	Нет ошибок
ErrCMD	0x01	Неверный/неизвестный код команды

Порядок следования бит: первый бит MSB.
Порядок следования байтов: старший байт первый.

Примечание – При чтении данных выдаваемые данные задерживаются на 1байт.

Команда чтения/записи считывает/записывает данные последовательно слово за словом вне зависимости от региона памяти, в который попадает адрес операции (буфер ПРМ, буфер ПРД, таблицы дескрипторов и т.д.).

При чтении данных из буфера данных ПРМ программист должен контролировать адрес операции и количество считываемых слов, чтобы избежать чтения данных из-за пределов буфера данных ПРМ.

Аналогичная процедура выполняется для операций записи буфера данных ПРМ и операций чтения/записи буфера данных ПРД.

Характеристики порта:
- независимые ПРМ и ПРД;
- скорость обмена до 20 Мбит/с.

5.3 Настройка контроллера

Настройка контроллера ЛВС необходима для правильной работы ПРМ и ПРД. Настройка контроллера может осуществляться в любое время – как сразу после завершения процедуры сброса, так и в процессе работы контроллера ЛВС. Часть настроек выполняется автоматически внутренними схемами управления и обеспечивает базовые параметры для осуществления приема и передачи пакетов. Для правильной работы контроллера требуется произвести дополнительные настройки.

По сбросу устанавливаются основные, часто используемые, настройки ПРМ и ПРД. ПРМ и ПРД после сброса выключены. Минимально необходимые действия по настройке включают указание MAC адреса контроллера ЛВС, настройка маски прерываний, включение ПРМ/ПРД.

5.3.1 Настройка режима приема/передачи пакетов

Устройство может работать в дуплексном или полудуплексном режимах на скорости передачи до 10 Мбит/с. По сбросу контроллер устанавливается в дуплексный режим работы.

5.3.2 Настройка работы в полудуплексном режиме

Полудуплексный режим работы контроллера включается установкой бита HALFD_EN=1 регистра MAC_CTRL. В этом режиме осуществляется контроль занятости линии (CRS) и коллизий (COL) не осуществляется.

В полудуплексном режиме пользователю доступны для установки следующие параметры:
- *окно коллизий**. Задаётся битами [7:0] регистра CollConfig и выражается в количестве тактов частоты TxCLK (2,5 МГц). При возникновении поздней коллизии модуль прекращает передачу пакета, выставляет состояние ошибки и переходит к обработке следующего пакета;
- *количество попыток передачи пакета*. Лимит количества попыток содержится в регистре CollConfig (биты [15:8]). Если количество коллизий, произошедших во время передачи пакетов, превышает значение этого регистра, модуль прекращает передачу пакета, выставляет состояние ошибки и переходит к обработке следующего пакета.

Примечание:

* Окно коллизий это время t после начала отправки пакета (в соответствии со спецификацией IEEE 802.3), по прошествии которого любая произошедшая коллизия считается поздней.

5.3.3 Настройка межпакетного интервала

Значение длины межпакетного интервала, выраженное в битовых интервалах (100 нс), заносится в регистр IPGTx. После отправки пакета передатчик блокирует обработку следующего дескриптора на заданное время.

5.3.4 Настройка размеров пакетов

Максимальный и минимальный размер пакета содержится в регистрах MaxFrame и MinFrame соответственно (в байтах).

Если размер пакета, передаваемый модулем, (в байтах) меньше значения регистра MinFrame, передатчик, если разрешено (бит PAD_DIS=0), дополняет поле данных пакета нулевыми байтами.

Количество нулевых байт N вычисляется как:

1) N = значение регистра MinFrame (байт) – L (длина пакета содержащаяся в поле длины дескриптора), если CRC_DIS = 1;
2) N = значение регистра MinFrame (байт) – L (длина пакета содержащаяся в поле длины дескриптора) – 4, если CRC_DIS = 0.

Если принимаемый пакет содержит количество байт больше значения, содержащегося в регистре MaxFrame, то приемник записывает в текущий дескриптор состояние ошибки и переходит к следующему дескриптору.

5.3.5 Расширенные настройки передачи пакетов

Контроллер ЛВС предоставляет возможность расширенного управления отправкой пакетов через поле управления дескриптора передачи.

Передаваемый пакет можно конфигурировать следующим образом:

- разрешить или запретить автоматическое добавление поля CRC в конец пакета (бит CRC_DIS дескриптора отсылаемых пакетов);
- разрешить или запретить автоматическое добавление нулевых бит к полям данных пакетов, имеющих длину поля данных меньше минимальной (бит PAD_DIS дескриптора отсылаемых пакетов).

5.3.6 Контроль MAC-адреса

Модуль MAC осуществляет автоматический контроль адреса назначения принимаемого пакета. Доступны следующие режимы фильтрации пакетов:

- прием всех пакетов;
- прием индивидуального пакета;
- прием широковещательных пакетов;
- прием пакетов по HASH таблице.

Режим «без фильтрации» (приём всех пакетов) осуществляется включением бита PRO_EN регистра MAC_CTRL.

Приём индивидуального пакета включен всегда и осуществляется по полному совпадению принятого в пакете MAC-адреса со значением, установленным в качестве MAC-адреса контроллера в регистрах MAC_ADDR_x.

Прием широковещательного пакета осуществляется при приема пакета с широковещательным MAC-адресом независимо от остальных режимов при включенном бите BCA_EN регистра MAC_CTRL.

Прием пакетов по HASH таблице осуществляется, если HASH-функция MAC-адреса принятого пакета соответствует маске в HASH-таблице контроллера. HASH-функция MAC-адреса вычисляется как «1» в позиции за номером, равным числу в старших 6 разрядах CRC32 MAC-адреса назначения (см. Приложение 6).

HASHTAG = 1<< ((CRC32(MAC))>>26);

Режим работает независимо от остальных и включается битом MCA_EN регистра MAC_CTRL.

5.4 Передача пакетов

Передача пакетов в контроллере ЛВС производится в 4 этапа:

1 этап: Копирование данных пакета УУ в буфер ПРД контроллера.

2 этап: Установка параметров передачи пакета записью соответствующих параметров в дескриптор ПРД пакета (см. приложение 2):

- формирование или нет прерываний по завершении обработки пакета (бит IRQ_EN);
- добавление или нет преамбулы (бит PRE_DIS);
- дополнение или нет пакета PAD-символами (0×00) до минимальной длины (бит PAD_DIS);
- дополнение или нет пакета полем CRC (бит CRC_DIS).

3 этап: Определение MAC-модулем по дескриптору пакета параметров передачи и собственно передача пакета. Передача включает автоматическую установку прочих параметров пакета, дополнение коротких пакетов до необходимой длины, расчет и вставка CRC, отработка временных интервалов при передаче пакетов, повторную отправку пакетов в случае ошибки при отправке.

4 этап: Формирование статусной информации о передаче, куда включаются признаки завершения обработки пакета (бит RDY), количество попыток отправки (поле RTRY), признак отказа от передачи из-за превышения лимита попыток (бит RL), признака ошибки в линии (бит LC). Также при разрешенных прерываниях передатчика выставляются прерывания завершения обработки пакета (бит TXF), наличия ошибок в пакете (бит TXE), передача пакета управления (бит TXC) и занятость передатчика по причине обслуживания контрольного пакета PAUSE (бит TX_BUSY).

По завершении передачи контроллер переходит к обработке очередного дескриптора из таблицы, согласно биту WRAP. При достижении конца таблицы производится переход к ее началу, независимо от бита WRAP последнего дескриптора.

Примечание – Бит RDY поля управления дескриптора должен устанавливаться последним(!).

Общий алгоритм передачи одиночного пакета приведен в Приложении 4.

5.5 Прием пакетов

Прием пакетов, как и передача, осуществляется в 4 этапа:

1 этап: Настройка дескриптора ПРМ (см. Приложение 3), которая заключается в установке признаков готовности дескриптора к обработке (бит RDY) и в установке признака разрешения формирования прерывания по завершении приема (бит IRQ_EN).

2 этап: Прием данных пакета. При приеме контролируются ошибки приема, и анализируется прочая информация, такая как MAC-адрес, с целью определения необходимости фиксации пакета в памяти и формировании статусной информации о приеме.

3 этап: Запись статусной информации в дескриптор:

- тип MAC-адреса принятого пакета;
- групповой пакет, прошедший проверку через HASH таблицу (бит MCA);
- широковещательный пакет (бит BCA);
- пакет с индивидуальным MAC-адресом (бит UCA);
- параметры пакета;

- пакет управления (бит CF);
- пакет длиной более MaxFrame (бит LF);
- пакет длиной менее MinFrame (бит SF);
- наличия ошибок при приёме пакета (бит EF);
- типы ошибок при приёме (если таковые были);
- несоответствие расчётного CRC пакета содержимому поля CRC (бит CRC_ERR);
- наличие ошибки в теле пакета (бит SMB_ERR);
- наличие ошибки типа LateCollision (бит LC);
- наличие переполнения буфера ПРМ при приёме пакета (бит OR).

А так же выставляются прерывания:

- завершения приема пакета (бит RXF);
- приема пакета с ошибками (бит RXE);
- прием пакета длиной более MaxFrame (бит RXL);
- прием пакета длиной менее MinFrame (бит RXS);
- прием пакета управления (бит RXC);
- переполнение буфера приёмника (бит RXBF_FULL);
- поступление пакета при неготовности текущего дескриптора (бит RXD_nREADY).

4 этап: По получении информации о приеме пакета (через бит занятости дескриптора или через прерывание), осуществляется считывание УУ пакета из памяти контроллера ЛВС.

По завершении передачи контроллер переходит к обработке очередного дескриптора из таблицы, согласно биту WRAP. При достижении конца таблицы производится переход к её началу, независимо от бита WRAP последнего дескриптора.

Примечание – Общий алгоритм приема одиночного пакета приведен в Приложении 4.

5.6 Работа PHY уровня

Модуль PHY контроллера ЛВС это одноканальный приемопередатчик протокола стандарта IEEE 802.3/Ethernet 10BASE-T, предоставляющий все функции протокола физического уровня.

Управление модулем осуществляется через регистры управления, находящиеся в общем пуле регистров.

Блок осуществляет контроль наличия коллизий в линии в полудуплексном режиме работы, а так же осуществляет функции слежения за подключением к линии по наличию передачи или импульсов LINK и формирования импульсных посылок LINK для индикации присутствия на линии. При превышении времени отсутствия изменений в линии блок формирует сигнал отсутствия подключения.

5.7 Индикация

В контроллере 5600ВГ1 предусмотрены выводы для индикации состояния при помощи внешних светодиодов.

Управление индикацией осуществляется через два вывода LED0 и LED1. Назначение выводов и их состояния приведены ниже (Таблица 6).

Частота мерцания индикаторов – 5 Гц.

Таблица 6 – Назначение и состояния выводов при индикации

Индикатор	Состояние	Назначение
LED0	0	Полудуплексный режим работы
LED0	1	Дуплексный режим работы
LED0	0→1→0→…	Наличие коллизии в линии
LED1	0	Соединение с линией
LED1	1	Отсутствует соединение с линией
LED1	0→1→0→…	Производится обмен пакетами

5.8 Прерывания

За обработку прерываний отвечают два регистра:
- регистр флагов прерываний;
- регистр маски прерываний.

Прерывания блокируются записью «0» в регистр маски прерываний.

При возникновении прерывания вывод INT устанавливается в низкий уровень. Вывод INT устанавливается в высокий уровень после очистки регистра флагов прерываний. Очистка регистра флагов прерываний осуществляется его чтением (при установленном бите GCTRL.READ_CLR_STAT) или записью «1» в соответствующий разряд регистра флагов INT_SRC.

Подробное описание флагов прерываний описано в Приложении 1.

5.9 Работа драйвера

Программное обеспечение (драйвер) предназначено для предоставления программисту средств управления контроллером. Драйвер включает в себя макроопределения регистровой модели устройства, определение регистровых структур и базовые функции для приема, отправки пакетов данных и проверки контроля результата их отправки.

Предоставляемая в драйвере процедура инициализации предназначена для инициализации контроллера ЛВС значениями, отличными от значений по умолчанию.

Предоставляемые функции отправки и приема пакета реализуют также и настройку соответствующего дескриптора по флагам, установленным в аргументах соответствующих функций.

Исходный код драйвера приведен в Приложении 5.

5.10 Особенности работы контроллера в разных режимах

5.10.1 Режим короткого замыкания (LoopBack)

Режим короткого замыкания (далее КЗ) предназначен для проведения проверок работы программно-аппаратной части устройства в пределах самого устройства без подключения к линии.

В контроллере ЛВС реализовано два способа организации режима КЗ:
- цифровое КЗ;
- аналоговое КЗ.

Цифровое КЗ предназначено для организации обратной петли на стыке MAC и PHY уровней. Режим устанавливается выставлением бита LB_EN регистра MAC_CTRL или выставлением бита DLB регистра PHY_CTRL.

Аналоговое КЗ служит для проверки работы и настройки всего контроллера в целом до подключения к линии. Режим устанавливается выставлением бита LB регистра PHY_CTRL.

5.10.2 Отключение от линии

Контроллер 5600ВГ1 может быть отключен от линии, как на прием, так и на передачу независимо друг от друга. Данный режим может быть использован при самотестировании работы контроллера программными средствами без нарушения работы линии. Включение и отключение ПРМ и ПРД осуществляется битами RXEN и TXEN соответственно (1 – включено, 0 – отключено). После сброса (программного или аппаратного) приемник и передатчик переводятся в неактивное состояние.

Отключение затрагивает только выходные каскады и не влияет на работу остальной части схемы. Таким образом, для работы в режиме КЗ нет необходимости в присутствии на линии прочих устройств.

6 Предельно-допустимые характеристики микросхемы

Таблица 7 – Предельно-допустимые и предельные режимы эксплуатации микросхем

Наименование параметра режима, единица измерения	Буквенное обозначение	Норма параметра			
		Предельно-допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Напряжение питания, В	U_{CC}	4,5	5,5	–	6,0
Входное напряжение низкого уровня, В, на выводах A[12:0], D[15:0], nCS, nOE, nWE, CLK, CLKS, nRST, JS_nTRST, SFS/TMS, SCLK/TCK, SDI/TDI, XS, X1	U_{IL}	0	0,8	минус 0,3	–
Входное напряжение высокого уровня, В, на выводе XS и X1 при XS=1	U_{IH}	0,7•U_{CC}	U_{CC}	–	U_{CC} +0,3
Входное напряжение высокого уровня, В, на выводах A[12:0], D[15:0], nCS, nOE, nWE, CLK, CLKS, nRST, JS_nTRST, SFS/TMS, SCLK/TCK, SDI/TDI	U_{IH}	2,4	U_{CC}	–	U_{CC} +0,3
Дифференциальное входное напряжение на выводах TPI+ и TPI-, мВ	U_{IT}	500	U_{CC}	–	–
Входное напряжение на выводах TPI+ и TPI-, В	U_I	–	–	минус 0,3	U_{CC} + 0,3
Ток нагрузки на цифровых выходах, мА, D[15:0], RDY, nIRQ, LED[1:0], SDO/TDO, X2	I_{OL} I_{OH}	минус 4,0	4,0	минус 10,0	10,0
Ток нагрузки на аналоговых выходах TPO++, TPO+, TPO--, TPO--, мА	I_{OL_A} I_{OH_A}	минус 40,0	40,0	минус 80,0	80,0
Частота следования импульсов тактовых сигналов, МГц, - в режиме обхода XS=1	f_C	–	80	–	–
- в режиме умножения частоты XS=0		–	10	–	–
Частота следования импульсов тактовых сигналов параллельного интерфейса, МГц, на выводе CLK	f_{CLK}*	–	40	–	–
Частота следования импульсов тактовых сигналов последовательного интерфейса, МГц, на выводе SCLK/TCK	$f_{SCLK/TCK}$*	–	10	–	–
Время нарастания и спада входного тактового сигнала, нс	t_{r_PLL}* t_{f_PLL}*	–	3	–	–
Минимальная длительность сигнала сброса, нс	t_{MCLR}*	100	–	–	–
Емкость нагрузки на выходах, пф	C_L	–	60	–	–

Примечания:

1 . *Временные параметры f_C, f_{CLK}, $f_{SCLK/TCK}$, t_{r_PLL}, t_{f_PLL}, t_{MCLR} гарантируются в процессе проведения ФК на максимальной частоте.

2. Не допускается одновременное задание нескольких предельных режимов.

7 Электрические параметры микросхемы

Таблица 8 – Электрические параметры микросхем при приёмке и поставке

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
Выходное напряжение низкого уровня на цифровых выходах D[15:0], RDY, nIRQ, LED[1:0], SDO/TDO, В	U_{OL}	–	0,4	25, 85, минус 60
Выходное напряжение высокого уровня на цифровых выходах D[15:0], RDY, nIRQ, LED[1:0], SDO/TDO, В	U_{OH}	4,05	–	25, 85, минус 60
Выходное напряжение низкого уровня на аналоговых выходах TPO++, TPO+, TPO--, TPO--, X2, В	U_{OL_A}	–	1,1	25, 85, минус 60
Выходное напряжение высокого уровня на аналоговых выходах TPO++, TPO+, TPO--, TPO--, X2, В	U_{OH_A}	3,5	–	25, 85, минус 60
Ток утечки низкого уровня на цифровых входах A[12:0], D[15:0], nCS, nOE, nWE, CLK, CLKS, nRST, JS_nTRST, SFS/TMS, SCLK/TCK, SDI/TDI, мкА	I_{ILL}	минус 1,0	1,0	25, 85, минус 60
Ток утечки высокого уровня на цифровых входах A[12:0], D[15:0], nCS, nOE, nWE, CLK, CLKS, nRST, JS_nTRST, SFS/TMS, SCLK/TCK, SDI/TDI, мкА	I_{ILH}	минус 1,0	1,0	25, 85, минус 60
Входной ток низкого уровня на аналоговых входах TPI+, TPI-, X1, мА	I_{IL_A}	минус 1,0	1,0	25, 85, минус 60
Входной ток высокого уровня на аналоговых входах TPI+, TPI-, X1, мА	I_{IH_A}	минус 1,0	1,0	25, 85, минус 60
Ток утечки низкого уровня на входе XS (13), мкА	I_{ILL_XS}	минус 50	50	25, 85, минус 60
Ток утечки высокого уровня на входе XS (13), мкА	I_{ILH_XS}	минус 50	50	25, 85, минус 60
Статический ток потребления, мА	I_{CC}	–	20	25, 85, минус 60
Статический ток потребления (приемопередатчики в состоянии «Выключено»), мкА	I_{CCS}	–	50	25, 85, минус 60
Динамический ток потребления, мА	I_{OCC}	–	190	25, 85, минус 60

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
Выходная частота генератора, МГц	f_{O_PLL}*	79,9	80,1	25, 85, минус 60
Параметры параллельного интерфейса				
Время цикла чтения/записи, нс	t_{CY}*	100	–	25, 85, минус 60
Время задержки при переходе выхода RDY из состояния «Выключено» в состояние высокого (низкого) уровня относительно сигнала nCS, нс	$t_{PZH(nCS-RDY)}$* $t_{PZL(nCS-RDY)}$*	–	12,5	25, 85, минус 60
Время задержки сигнала RDY при переходе выхода из состояния высокого (низкого) уровня в состояние «Выключено» относительно сигнала nCS, нс	$t_{PHZ(nCS-RDY)}$* $t_{PLZ\,(nCS-RDY)}$*	–	12,5	25, 85, минус 60
Время задержки установки сигнала RDY относительно сигнала nCS, нс	$t_{DH(nCS-RDY)}$*	87,5	–	25, 85, минус 60
Время задержки снятия сигнала RDY относительно nCS, нс, при: CLKS=0	$t_{DL(nCS-RDY)}$*	–	H**	25, 85, минус 60
при: CLKS=1		–	12,5	
Время выборки данных относительно сигнала nOE, в цикле чтения, нс	$t_{AR(nOE-D)}$*	–	112,5	25, 85, минус 60
Время удержания данных относительно сигнала nOE, в цикле чтения, нс	$t_{HR(nOE-D)}$*	12,5	–	25, 85, минус 60
Время предустановки данных относительно начала сигнала разрешения записи nWE, в цикле записи, нс при: CLKS=0	$t_{SUW0(D-nWE)}$*	12,5	–	25, 85, минус 60
Время удержания данных относительно сигнала разрешения записи nWE, в цикле записи, нс при: CLKS=0	$t_{HW0(nWE-D)}$*	12,5	–	25, 85, минус 60
Параметры последовательного интерфейса				
Время задержки сигнала SDI (SDO) относительно SCLK, нс	$t_{D(SCLK-SDI)}$* $t_{D(SCLK-SDO)}$*	12,5	–	25, 85, минус 60
Время предустановки сигнала SFS относительно сигнала SCLK, нс	$t_{SU(SFS-SCLK)}$*	12,5	–	25, 85, минус 60
Время предустановки данных относительно сигнала SCLK, нс	$t_{SU(SDI-SCLK)}$* $t_{SU(SDO-SCLK)}$*	12,5	–	25, 85, минус 60

Наименование параметра, единица измерения, режим измерения	**Буквенное обозначение параметра**	**Норма параметра**		**Температура среды, °С**
		не менее	**не более**	
Время удержания сигнала SFS относительно сигнала SCLK, нс	$t_{H(SCLK-SFS)}$*	12,5	–	25, 85, минус 60

Примечания:

1. *Временные параметры f_{O_PLL}, t_{CY}, $t_{PZH(nCS-RDY)}$, $t_{PZL(nCS-RDY)}$, $t_{PHZ(nCS-RDY)}$, $t_{PLZ\ (nCS-RDY)}$, $t_{DH(nCS-RDY)}$, $t_{DL(nCS-RDY)}$, $t_{AR(nOE-D)}$, $t_{HR(nOE-D)}$, $t_{SUW0(D-nWE)}$, $t_{HW0(nWE-D)}$, $t_{D(SCLK-SDI)}$, $t_{D(SCLK-SDO)}$, $t_{SU(SFS-SCLK)}$, $t_{SU(SDI-SCLK)}$, $t_{SU(SDO-SCLK)}$, $t_{H(SCLK-SFS)}$ гарантируются в процессе проведения ФК на максимальной частоте.

2. ** $H = 1 / 2 \bullet f_{CLK}$

8 Временные диаграммы

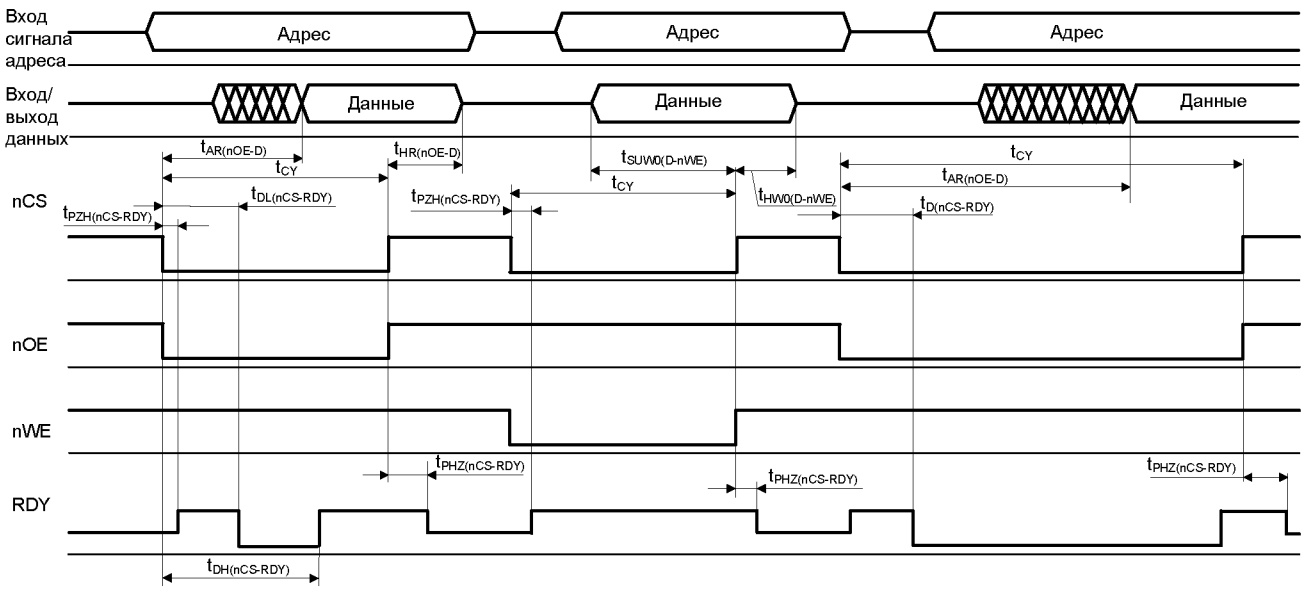


Рисунок 8 – Временная диаграмма циклов чтения и записи параллельного интерфейса при: CLKS=0, CLKS=1

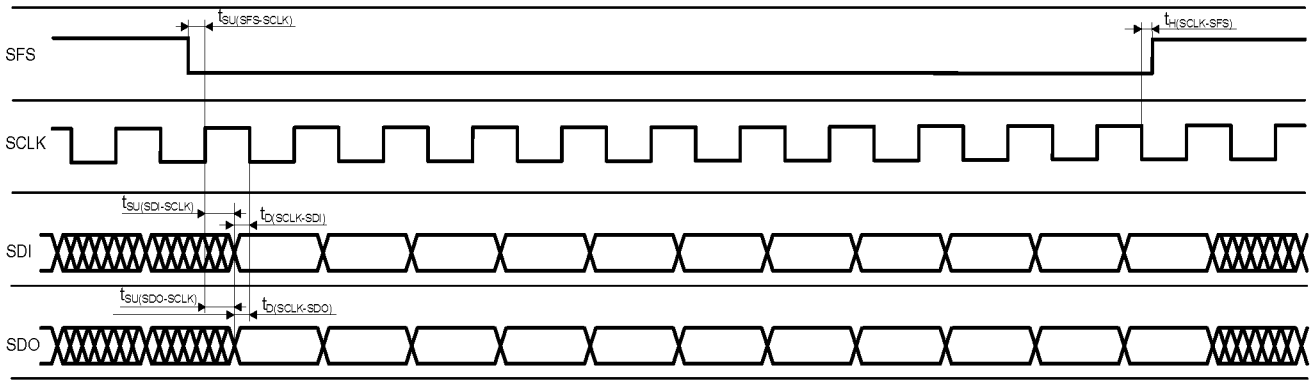
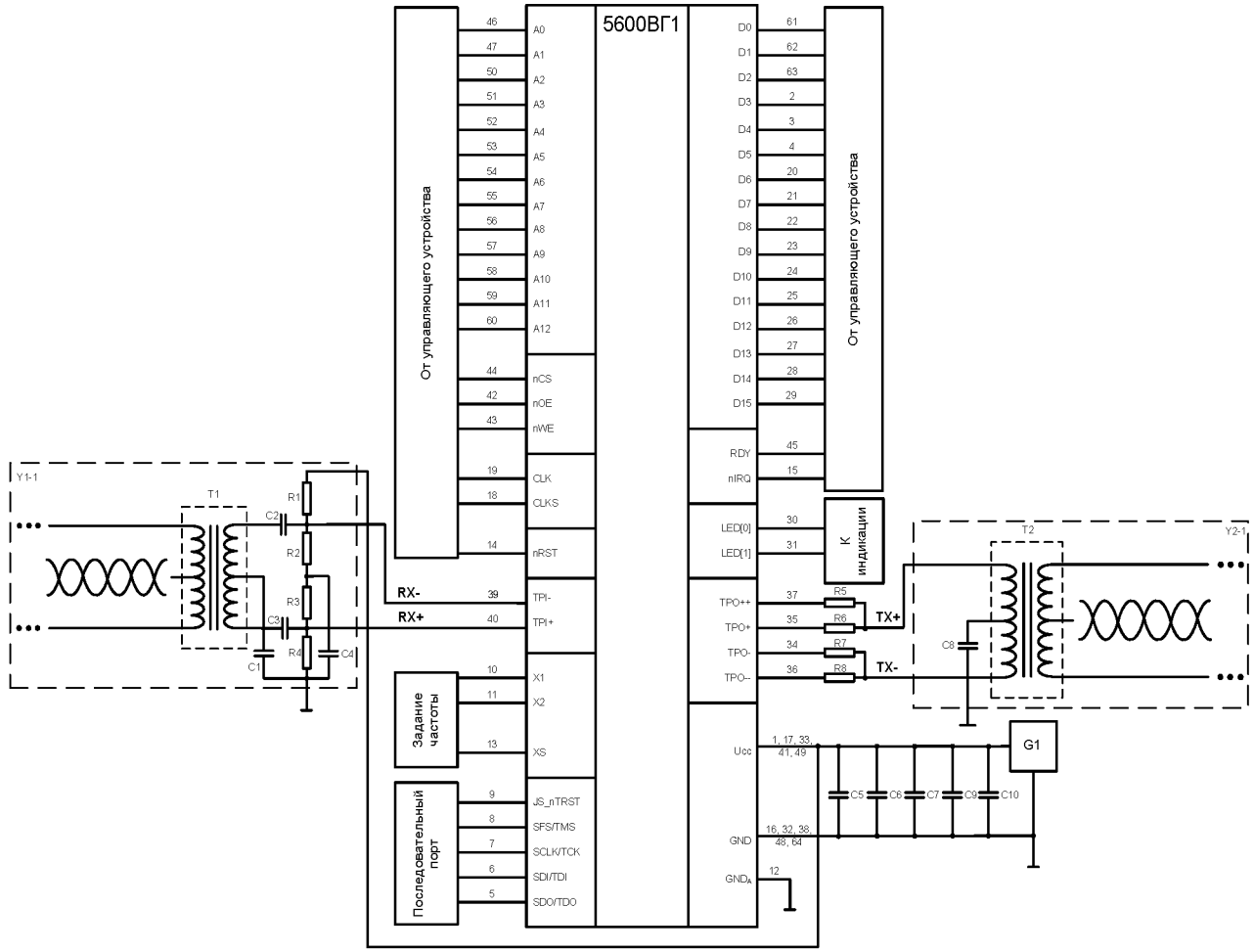


Рисунок 9 – Временная диаграмма циклов чтения и записи последовательного интерфейса

9 Схема подключения



5600ВГ1 – включаемая микросхема;
G2 – источник постоянного напряжения, U_{CC} = 5,0 В;
R1 – R8 – сопротивления;

Для линии с волновым сопротивлением 100 Ом:
R1 = R4 = 1 кОм* ± 5 %, 0,25 Вт;
R2 = R3 = 50 Ом ± 5 %, 0,25 Вт;
R5 = R8 = 390 Ом ± 5 %, 0,25 Вт;
R6 = R7 = 47 Ом ± 5 %, 0,25 Вт.

Для линии с волновым сопротивлением 75 Ом:
R1 = R4 = 1 кОм* ± 5 %, 0,25 Вт;
R2 = R3 = 36 Ом ± 5 %, 0,25 Вт;
R5 = R8 = 270 Ом ± 5 %, 0,25 Вт;
R6 = R7 = 36 Ом ± 5 %, 0,25 Вт.

Рисунок 10 – Схема реализации устройства на базе одной микросхемы

Продолжение к рисунку 10

C1–C9 – конденсаторы;
C1 = C4 = C5 = C6 = C7 = C8 = C9 = C10 = 0,1 мкФ ± 10 %;
C2 = C3 = не более 33 мкФ и не менее 33 нФ.
T1, T2 – трансформаторы, L = 350 мкГн, 1:1;
Y1 – элементы схемы (Y1-1; – Y1-2).

* – Резисторы R1 и R4 могут не устанавливаться, при этом не гарантируется прием LINK-импульсов и, как следствие, − невозможность определения подключения или отключения линии.

Номиналы резисторов подбираются согласно волновому сопротивлению провода и равны 1/2 • R провода.

10 Справочные данные

Максимальное значение емкости вывода микросхемы на частотах менее 1 МГц не превышает 9 пФ.

Таблица 9 – Справочные параметры

Наименование параметра, единица измерения, режим измерения	**Буквенное обозначение параметра**	**Норма параметра**		**Температура среды, °С**
		не менее	**не более**	
Входное сопротивление приемника, кОм по выводам	R_{I_RCV}	5	18	25, 85, минус 60
Длительность сигнала высокого уровня (импульса LINK), нс, при U_{CC}= 4,5 В	t_{WHWT}	100	–	25, 85, минус 60
Период выдачи импульса LINK, мс, при U_{CC}= 4,5 В	T_{WHWT} T_{WHUD}	1	64	25, 85, минус 60

11 Типовые зависимости

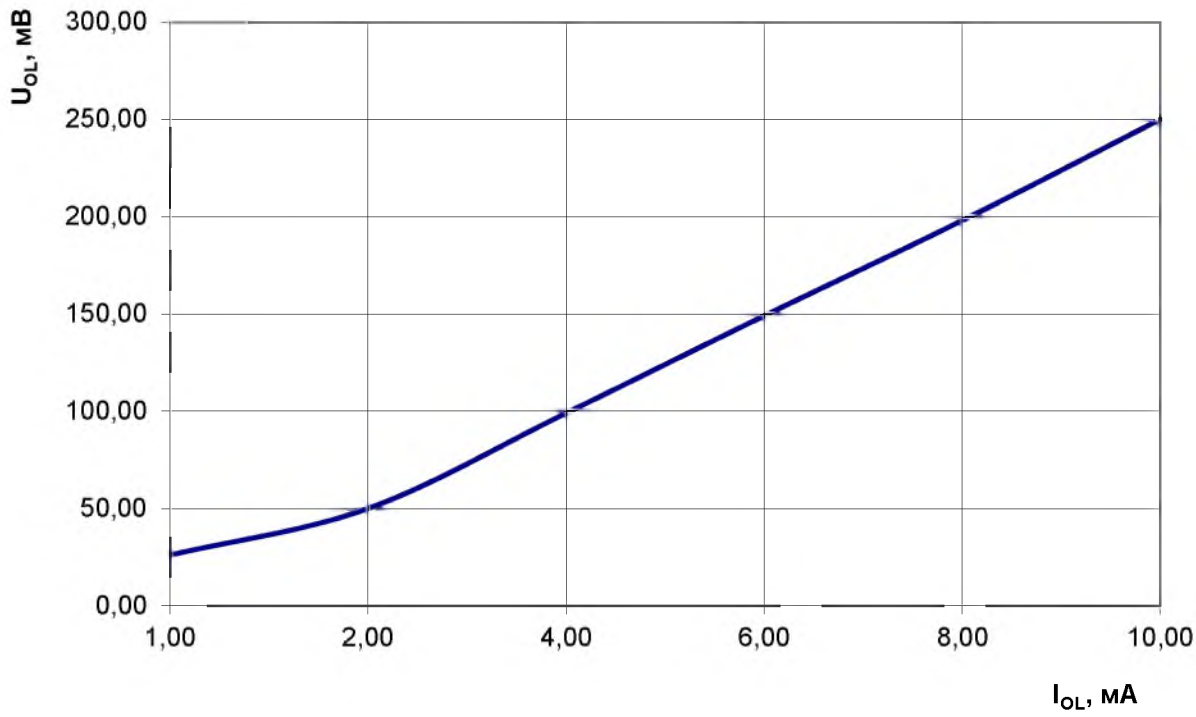


Рисунок 11 – Зависимость выходного напряжения низкого уровня на цифровых выходах D[15:0], RDY, nIRQ, LED[1:0], SDO/TDO U_{OL} от тока нагрузки на цифровых выходах I_{OL}

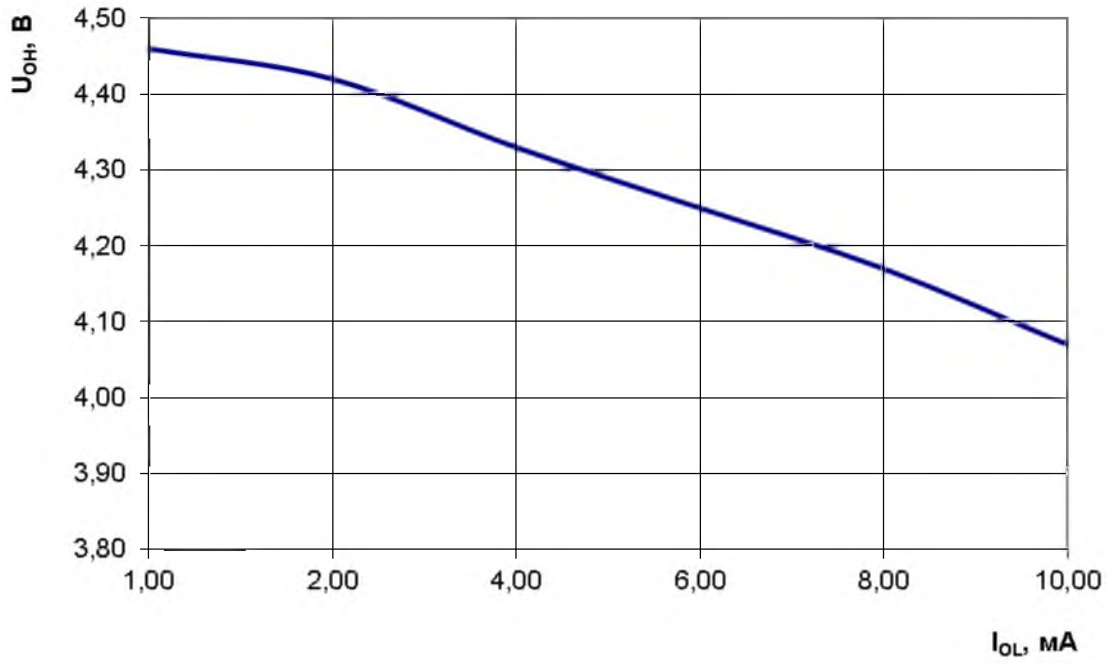


Рисунок 12 – Зависимость выходного напряжения высокого уровня на цифровых выходах D[15:0], RDY, nIRQ, LED[1:0], SDO/TDO U_{OH} от тока нагрузки на цифровых выходах I_{OL}

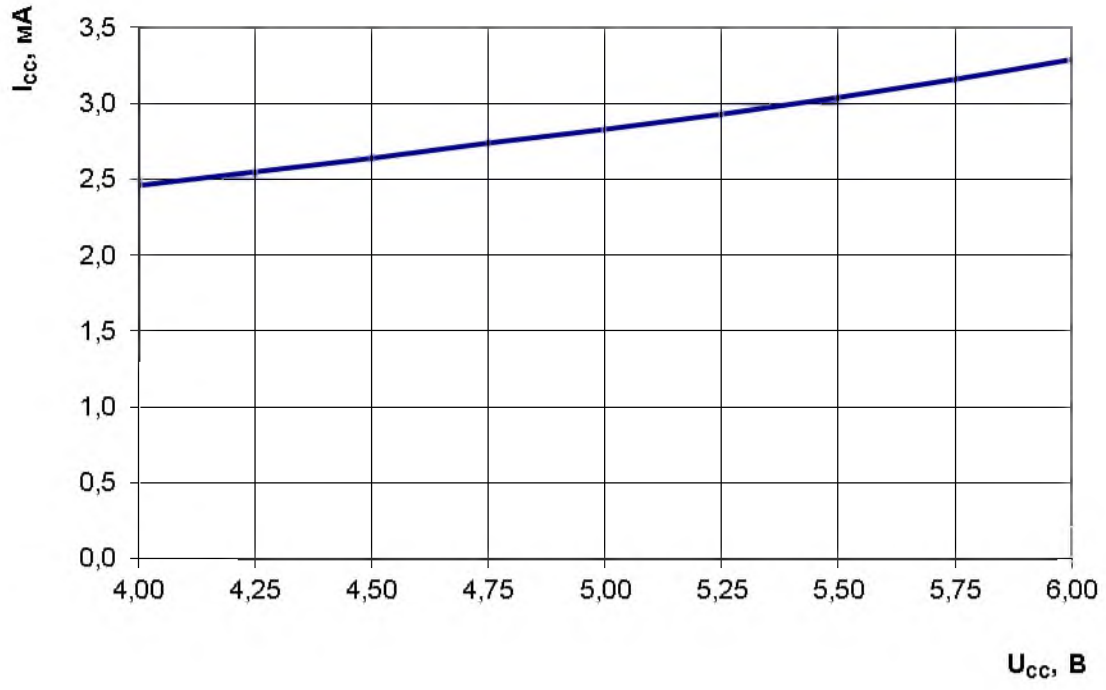


Рисунок 13 – Зависимость статического тока потребления I_{CC} от напряжения питания U_{CC}

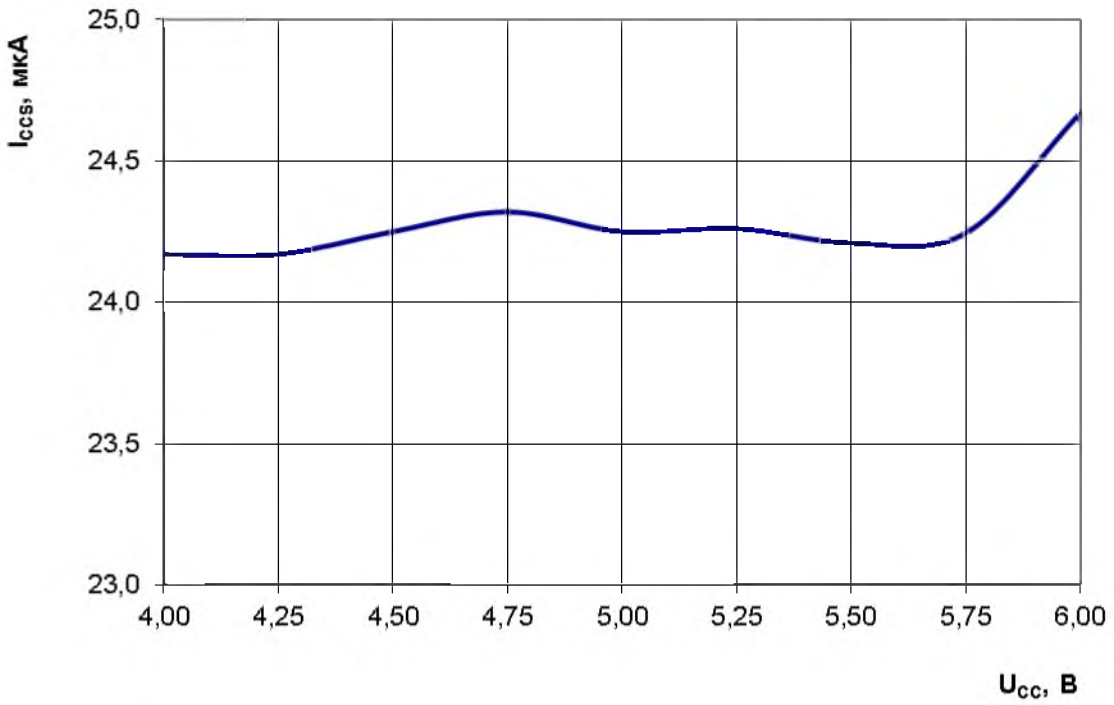


Рисунок 14 – Зависимость статического тока потребления (приемопередатчики в состоянии «Выключено») I_{CCS} от напряжения питания U_{CC}

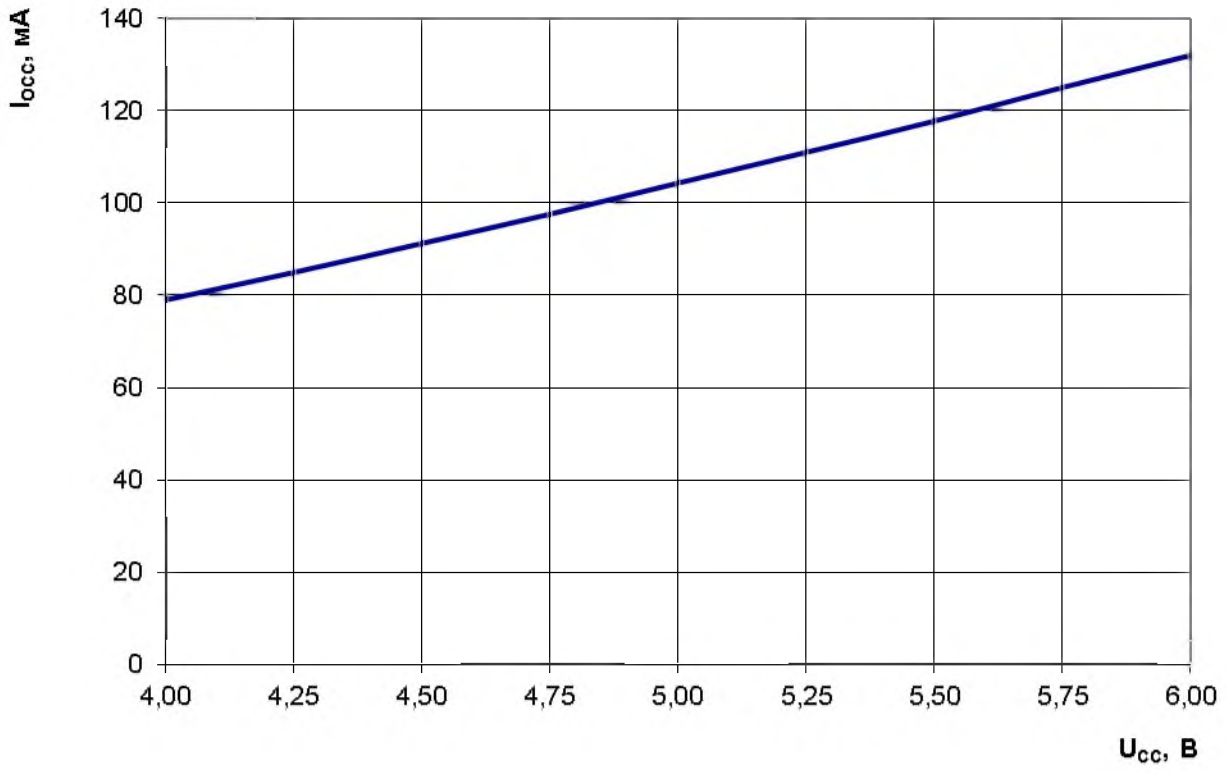


Рисунок 15 – Зависимость динамического тока потребления I_{OCC} от напряжения питания U_{CC}

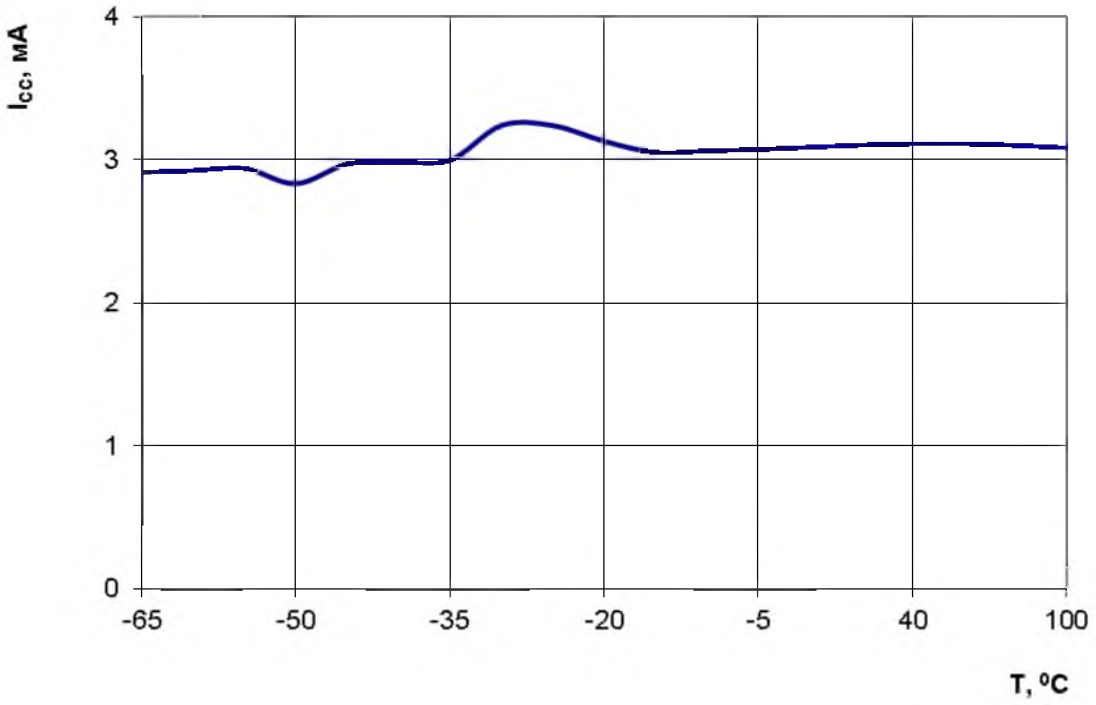


Рисунок 16 – Зависимость статического тока потребления I_{CC} от температуры T°

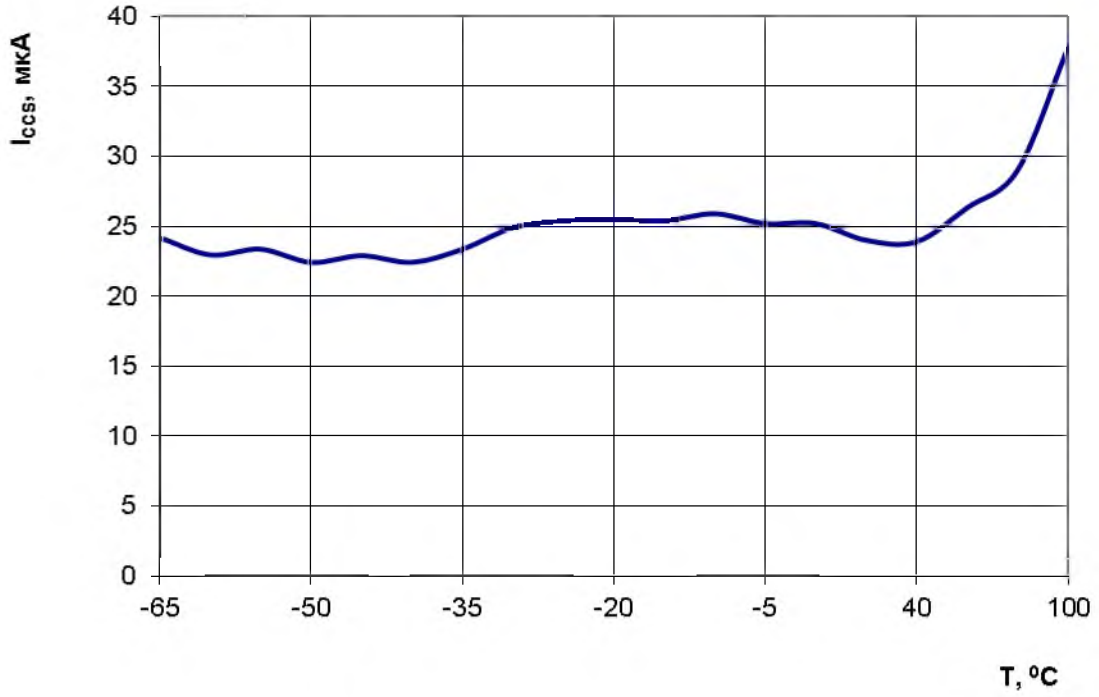


Рисунок 17 – Зависимость статического тока потребления (приемопередатчики в состоянии «Выключено») Iocc от температуры T°

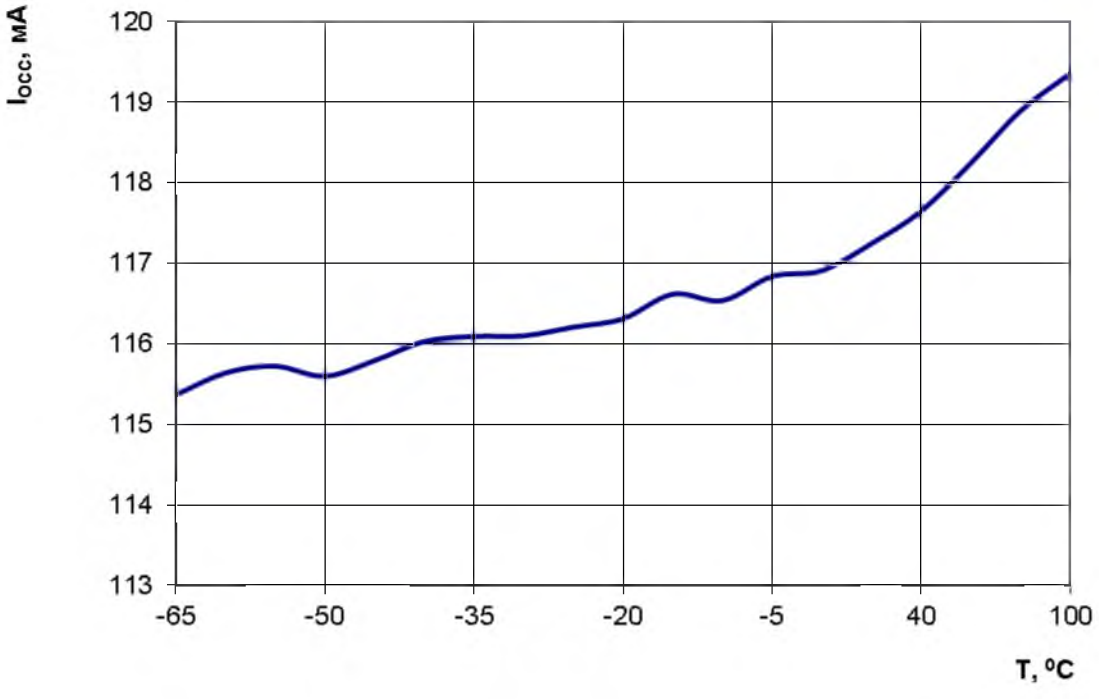


Рисунок 18 – Зависимость динамического тока потребления Iocc от температуры T°

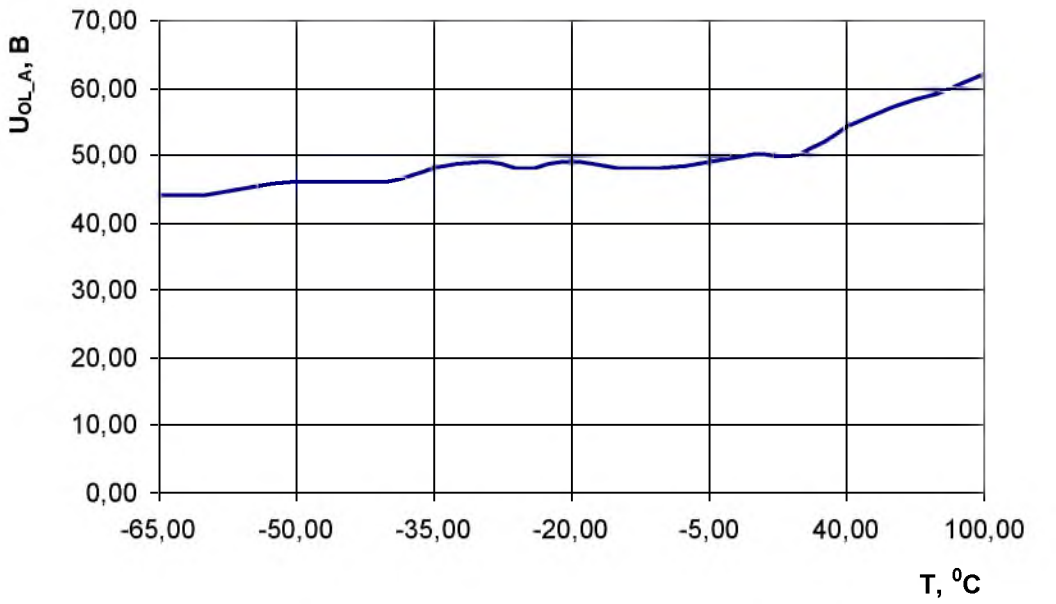


Рисунок 19 – Зависимость выходного напряжения низкого уровня на аналоговых выходах ТРО++, ТРО+, ТРО--, ТРО--, X2 от температуры Т°, при I_{OL_A} = 6 мА

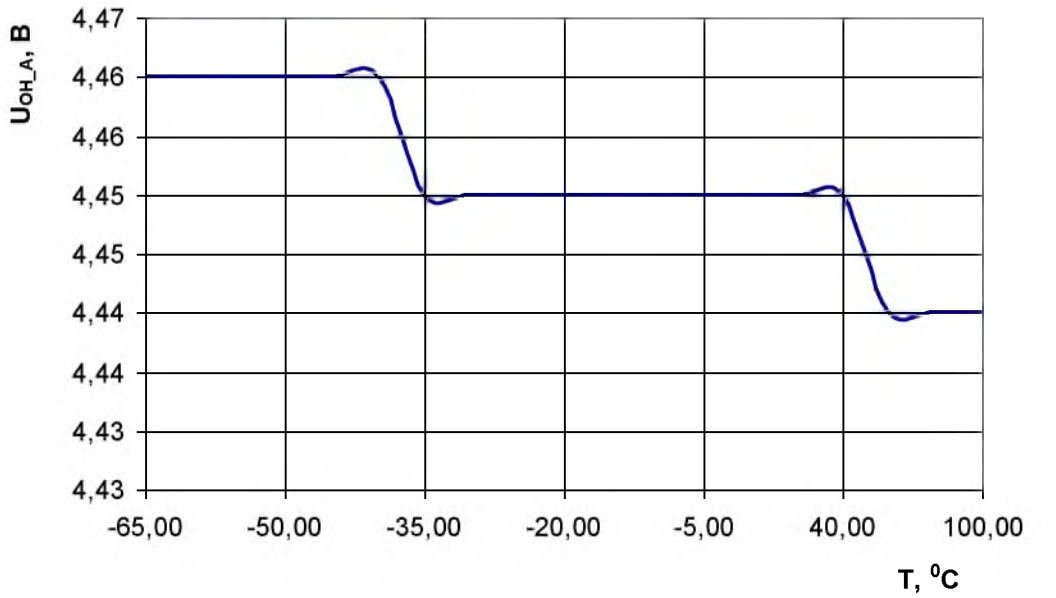


Рисунок 20 – Зависимость выходного напряжения высокого уровня на аналоговых выходах ТРО++, ТРО+, ТРО--, ТРО--, X2 от температуры Т°, при I_{OL_A} = 6 мА

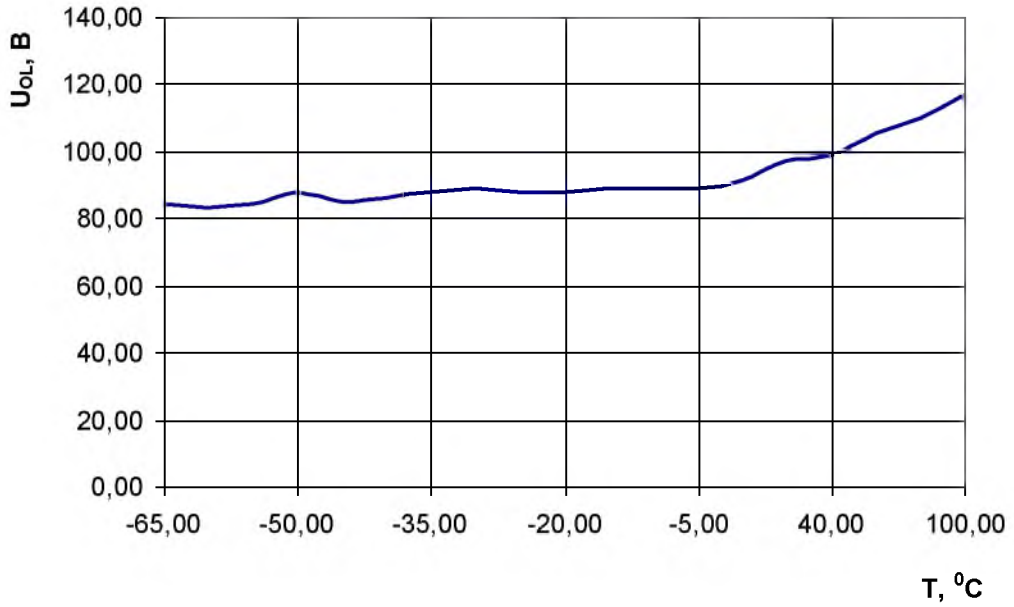


Рисунок 21 – Зависимость выходного напряжения низкого уровня на цифровых выходах D[15:0], RDY, nIRQ, LED[1:0], SDO/TDO от температуры T°, при I_{OL} = 4 мА

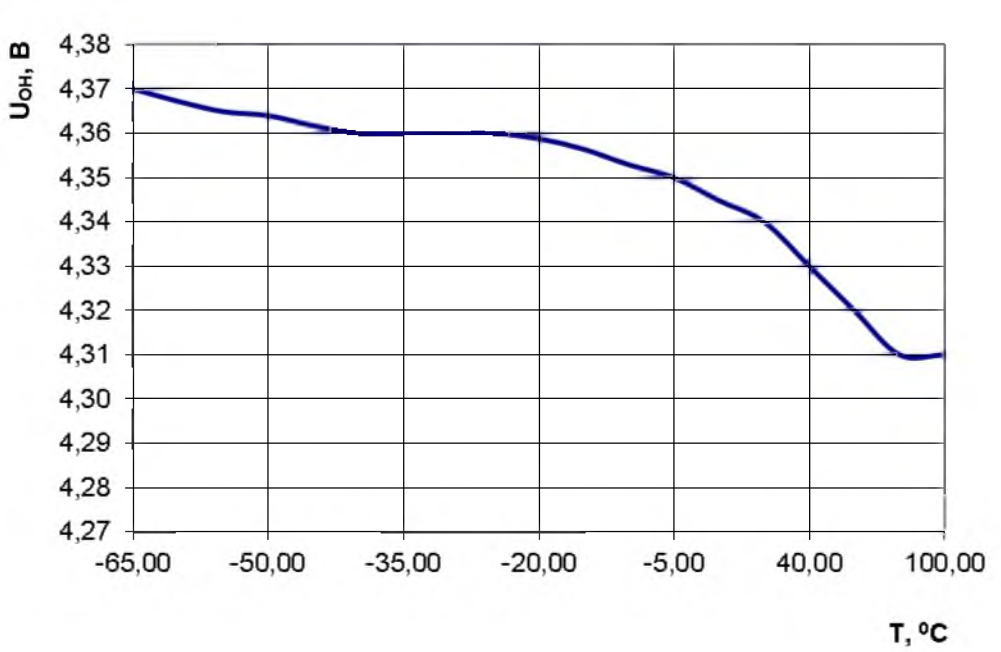


Рисунок 22 – Зависимость выходного напряжения высокого на цифровых выходах D[15:0], RDY, nIRQ, LED[1:0], SDO/TDO от температуры T°, при I_{OL} = 4 мА

12 Габаритный чертеж микросхемы

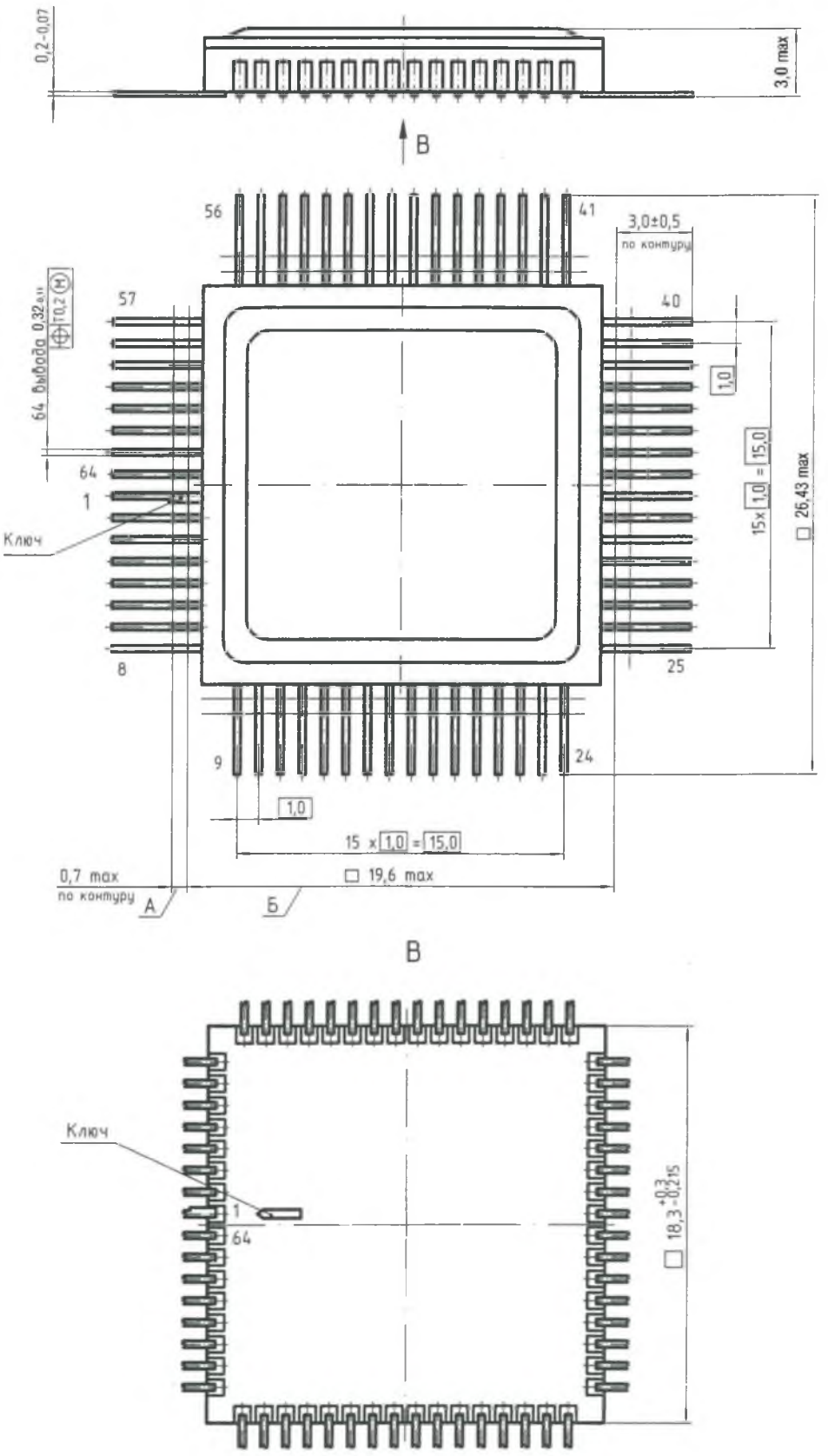


Рисунок 23 – Корпус H18.64-1B

13 Информация для заказа

Обозначение микросхемы	Маркировка	Тип корпуса	Температурный диапазон
5600ВГ1У	5600ВГ1У	Н18.64-1В	минус 60 – 85 °С
К5600ВГ1У	К5600ВГ1У	Н18.64-1В	минус 60 – 85 °С
К5600ВГ1УК	К5600ВГ1У•	Н18.64-1В	0 – 70 °С

Микросхемы с приемкой «ВП» маркируются ромбом.
Микросхемы с приемкой «ОТК» маркируются буквой «К».

ПРИЛОЖЕНИЕ 1. Регистры контроллера

Таблица 10 – Регистры контроллера

Наименование	Доступ	Номер регистра	Значение по умолчанию	Описание
MAC_CTRL	RW	0x00	0xC0A0	Регистр управления MAC уровнем контроллера
MinFrame	RW	0x01	0x0040	Регистр задания минимальной длины пакета
MaxFrame	RW	0x02	0x0600	Регистр задания максимальной длины пакета
CollConfig	RW	0x03	0x0000	Регистр управления обработкой коллизий
IPGTx	RW	0x04	0x0006	Регистр задания межпакетного интервала (в битовых интервалах = 100 нс)
MAC_ADDR_T	RW	0x05	0x89AB	Регистр младшей части MAC-адреса
MAC_ADDR_M	RW	0x06	0x4567	Регистр средней части MAC-адреса
MAC_ADDR_H	RW	0x07	0x0123	Регистр старшей части MAC-адреса
HASH0	RW	0x08	0x0000	Регистр слова 0 HASH таблицы
HASH1	RW	0x09	0x0000	Регистр слова 1 HASH таблицы
HASH2	RW	0x0A	0x0000	Регистр слова 2 HASH таблицы
HASH3	RW	0x0B	0x8000	Регистр слова 3 HASH таблицы
INT_MSK	RW	0x0C	0x0000	Регистр маски прерываний
INT_SRC	RW	0x0D	0x0000	Регистр флагов прерываний
PHY_CTRL	RW	0x0E	0x81D0	Регистр управления PHY уровнем контроллера
PHY_STAT	R	0x0F	0xXXXX	Регистр состояния PHY уровня
RXBF_HEAD	RW	0x10	0x07FF	Регистр начала данных буфера приемника (управляется программистом и указывает на последний свободный адрес в буфере)
RXBF_TAIL	R	0x11	0x0000	Регистр конца данных буфера приемника (управляется контроллером и указывает на первый свободный адрес в буфере)
		0x12..0x13		Зарезервировано
STAT_RX_ALL	R	0x14	0x0000	Счетчик количества входящих пакетов (дошедших до MAC-уровня)
STAT_RX_OK	R	0x15	0x0000	Счетчик количества успешно принятых входящих пакетов
STAT_RX_OVF	R	0x16	0x0000	Счетчик количества входящих пакетов, вызвавших переполнение буфера ПРМ
STAT_RX_LOST	R	0x17	0x0000	Счетчик количества входящих пакетов, потерянных из-за неготовности MAC-уровня к приему (неготовность дескриптора)
STAT_TX_ALL	R	0x18	0x0000	Счетчик количества исходящих пакетов
STAT_TX_OK	R	0x19	0x0000	Счетчик количества успешно отосланных исходящих пакетов
		0x1A..0x1E		Зарезервировано
GCTRL	RW	0x1F	0x4382	Регистр управления стыка HOST-контроллер

Примечание – Адрес регистра в адресном пространстве внешней шины определяется как <номер регистра> + 0x1FC0.
Значение резервных битов при чтении не определено.
Поведение при записи по зарезервированным адресам регистров не определено.

GCTRL

Таблица 11 – Описание бит регистра GCTRL

Обозначение	Бит	Описание
GLBL_RST	15	Общий сброс всего контроллера 1 – сброшен; 0 – рабочее состояние
READ_CLR_STAT	14	Очистка статистики по чтению 1 – при чтении регистров/флагов состояния они очищаются; 0 – для очистки статусов необходима запись
SPI_RST	13	Сброс встроенного контроллера последовательного порта 1 – сброшен; 0 – рабочее состояние
ASYNC_MODE	12	Переключение между режима формирования сигнала RDY 1 – асинхронный; 0 – синхронный
	11..0	Зарезервировано

MAC_CTRL

Таблица 12 – Описание бит регистра MAC_CTRL

Обозначение	Бит	Описание
TX_RST	15	Сброс ПРД MAC-уровня 1 – сброшен; 0 – рабочее состояние
RX_RST	14	Сброс ПРМ MAC-уровня 1 – сброшен; 0 – рабочее состояние
-	13	Зарезервировано
DSCR_SCAN_EN	12	Переключение ПРД в режим сканирования дескрипторов, когда переход к следующему осуществляется вне зависимости от готовности 0 – при неготовности бита RDY в поле управления ожидание; 1 – переход к следующему дескриптору
PAUSE_EN	11	Разрешение обработки Pause Frame 1 – пакет PAUSE обрабатывается автоматически
PRO_EN	10	Включение приема всех пакетов независимо от их MAC-адреса 1 – включен; 0 – выключен
BCA_EN	9	Включение приема всех широковещательных пакетов 1 – включен; 0 – выключен
MCA_EN	8	Включение приема всех пакетов, соответствующих HASH-таблице

Обозначение	Бит	Описание
		1 – включен; 0 – выключен
CTRL_FRAME_EN	7	Разрешение приема управляющих пакетов 1 – включен; 0 – выключен
LONG_FRAME_EN	6	Разрешение приема пакетов длиной более MaxFrame 1 – разрешен; 0 – не разрешен
SHRT_FRAME_EN	5	Разрешение приема пакетов длиной менее MinFrame 1 – разрешен; 0 – не разрешен
ERR_FRAME_EN	4	Разрешение приёма пакетов с ошибками 1 – разрешен; 0 – не разрешен
BCKOF_DIS	3	Отключение интервала ожидания перед повторением отправки пакета в случае коллизии 1 – включен; 0 – выключен
HALFD_EN	2	Переключение в режим полудуплексного приёма-передачи 1 – включен; 0 – выключен
BIG_ENDIAN	1	Переключение в режим BIG ENDIAN формата данных 1 – включен; 0 – выключен
LB_EN	0	Включение тестового замыкания ПРД на ПРМ м/у уровнями MAC и PHY 1 – включен; 0 – выключен

COLLCONF

Таблица 13 – Описание бит регистра COLLCONF

Обозначение	Бит	Описание
-	15…12	Зарезервировано
RetriesLimit	11...8	Максимальное возможное количество попыток повтора передачи пакета (0 … 15)
CollisionWindow	7…0	Размер окна разрешенных коллизий. Если коллизия случается в пределах данного окна – осуществляется повтор передачи пакета. В противном случае сообщается об ошибке в линии. Задается в 4-битовых интервалах (400 нс). 0 = 0 нс 1 = 400 нс … 255 = ~100 мкс

IPGTx

Таблица 14 – Описание бит регистра IPGTx

Обозначение	Бит	Описание
IPG	15...0	Минимальный интервал отправки пакетов. Задается в битовых интервалах (100 нс). 0 = 0 нс 1 = 100 нс … 2^{16} = ~6 мс

INT_MSK/INT_SRC

Таблица 15 – Описание бит регистра INT_MSK/ INT_SRC

Обозначение	Бит	Описание
RXF	15	Индикатор успешного приема пакета
RXE	14	Индикатор наличия ошибок при приеме пакета
RXC	13	Индикатор приема пакета управления
RXBF_FULL	12	Индикатор переполнения буфера ПРМ при приеме пакета
RXD_nREADY	11	Индикатор неготовности текущего дескриптора в ПРМ
RXL	10	Индикатор приема пакета длиной более MaxFrame
RXS	9	Индикатор приема пакета длиной менее MinFrame
TXF	7	Индикатор успешной передачи пакета
TXE	6	Индикатор наличия ошибок при передаче пакета
TXC	5	Индикатор передачи пакета управления
	4...1	Зарезервировано
TX_BUSY	0	Индикатор принятия и обслуживания пакета Pause

Примечания:

INT_SRC – регистр флагов прерываний (1 – наличие соответствующего события). Снятие флагов осуществляется записью «1» в INT_SRC. Либо чтением при установленном бите GCRTL.READ_CLR_STAT = 1.

INT_MSK – регистр маски прерываний (0 – запрещено, 1 – разрешено)

PHY_CTRL

Таблица 16 – Описание бит регистра PHY_CTRL

Обозначение	Бит	Описание
RST	15	Сброс встроенного контроллера PHY-уровня 1 – состояние сброса; 0 – рабочее состояние
-	14	Переключение на работу с внешним контроллером PHY-уровня. 0 – используется внутренний PHY; 1 – используется внешний PHY. Интегрированный PHY в сбросе (в текущей ревизии выводы для подключения внешнего PHY отсутствуют)

Обозначение	Бит	Описание
TXEN	13	Разрешение работы ПРД встроенного контроллера PHY-уровня 1 – ПРД активен; 0 – отключен
RXEN	12	Разрешение работы ПРМ встроенного контроллера PHY-уровня 1 – ПРМ активен; 0 – отключен
LINK_PERIOD	11:6	Период следования LINK импульсов в мс +2мс. Период ожидания LINK импульсов вдвое больше заданного периода следования LINK импульсов (диапазон 2...64 мс)
BASE_2	5	Переключение на работу с коаксиальным кабелем в режиме полудуплексного приема/передачи 1 – подключение по коаксиальному кабелю; 0 – подключение по витой паре
DIR	4	Порядок передачи битов в полубайте 1 – MSB; 0 – LSB
EARLY_DV	3	Включение формирования сигнала RxDV одновременно с сигналом CRS 0– штатный, сигнал RxDV формируется вместе с первыми значащими битами пакета (после приема поля SFD); 1– ранний, сигнал RxDV формируется вместе с битами пакета
HALFD	2	Включение режима полудуплексного приема-передачи 1 – полудуплексный режим; 0 – дуплексный режим
DLB	1	Включение тестового замыкания ПРД на ПРМ на входе контроллера PHY-уровня 1 – режим замыкания входов ПРД на выход ПРМ; 0 – штатный режим
LB	0	Включение тестового замыкания ПРД на ПРМ на выходе контроллера PHY-уровня до аналоговой части ПРМ/ПРД 1 – режим замыкания выходов ПРД на вход ПРМ; 0 – штатный режим

PHY_STAT

Таблица 17 – Описание бит регистра PHY_STAT

Обозначение	Бит	Описание
	15..11	Зарезервировано
LINK	10	Индикатор встроенного контроллера PHY-уровня о наличии подключения в линии 0 – есть подключение; 1 – подключение отсутствует
	9..0	Зарезервировано

STAT_RX_ALL, STAT_RX_OK, STAT_RX_OVF, STAT_RX_LOST, STAT_TX_ALL, STAT_TX_OK

Счетчики пакетов. Отсчет ведется циклически. В регистрах STAT_RX_OK, STAT_RX_OVF, STAT_TX_ALL и STAT_TX_OK отсчет ведется при установленном бите IRQ_EN в поле управления соответствующего дескриптора. Очистка производится при чтении (при GCTRL.READ_CLR_STAT = 1) или записью 0 в соответствующий регистр.

ПРИЛОЖЕНИЕ 2. Дескриптор отсылаемых пакетов

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RDY	WRAP	IRQ_EN			PRE_DIS	PAD_DIS	CRC_DIS	RC[2:0]				RL	LC		CS
LEN [15:0] (Packet Length)															
(reserved)															
ADDR [15:0] (Packet Start Address)															

Примечание – Значащие быты адреса пакета 10...0.

Таблица 18 – Биты словосостояния дескриптора ПРД

Обозначение	Бит	Описание
RDY	15	Индикатор состояния исходящего пакета 1 – состояние готовности; 0 – отправлен/не заполнен
WRAP	14	Индикатор последнего дескриптора в таблице 1 – переход к дескриптору #0
IRQ_EN	13	Разрешение формирования прерываний по передаче 1 – разрешено; 0 – не разрешено
	12..11	Зарезервировано
PRE_DIS	10	Отключение передачи преамбулы 1 – передача преамбулы отключена; 0 – стандартный пакет
PAD_DIS	9	Отключение дополнения пакетов длиной менее MinFrame до минимальной длины PAD- символами (0×00) 1 – дополнение не производится; 0 – дополнение производится
CRC_DIS	8	Отключение дополнения пакетов полем CRC32 1 – дополнение не производится; 0 – дополнение производится
RTRY	7…4	Счетчик количества попыток передачи исходящего пакета
RL	3	Индикатор использования разрешенного количества повторений в случае неуспешной отправки исходящего пакета 1 – за установленное число попыток пакет передать не удалось
LC	2	Индикатор наличия Late Collision 1 – обнаружен сигнал COL за пределами «окна коллизий»
	1	Зарезервировано
CS	0	Индикатор потери несущей во время передачи пакета 1 – имеется потеря сигнала CRS в процессе передачи

ПРИЛОЖЕНИЕ 3. Дескриптор принимаемых пакетов

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RDY	WRAP	IRQ_EN			MCA	BCA	UCA	CF	LF	SF	EF	CRC_ERR	SMB_ERR		OR
LEN [15:0] (Packet Length)															
(reserved)															
ADDR [15:0] (Packet Start Address)															

Примечание – Значащие быты адреса пакета 10..0.

Таблица 19 – Биты словосостояния дескриптора ПРМ

Обозначение	Бит	Описание
RDY	15	Индикатор состояния исходящего пакета 1 – готов к приему пакета; 0– пакет принят/не готов
WRAP	14	Индикатор последнего дескриптора в таблице 1– переход к дескриптору #0
IRQ_EN	13	Разрешение формирования прерываний по передаче 1 – разрешено; 0 – не разрешено
	12..11	Зарезервировано
MCA	10	Индикатор приема группового пакета с MAC-адресом соответствующего HASH-таблице 1 – совпадение адреса HASH-таблице
BCA	9	Индикатор приема широковещательного пакета 1 – принят широковещательный пакет
UCA	8	Индикатор приема индивидуального пакета с полным совпадением MAC-адреса 1 – точное совпадение адреса
CF	7	Индикатор приема пакета управления 1 – принят пакет управления
LF	6	Индикатор приёма пакета длиной более MaxFrame 1 – принят пакет длиннее MaxFrame
SF	5	Индикатор приема пакета длиной менее MinFrame 1 – принят пакет короче MinFrame
EF	4	Индикатор наличия ошибок при приеме пакета (сводный бит по ошибкам)
CRC_ERR	3	Индикатор наличия ошибки CRC при приёме пакета (1 – ошибка проверки CRC)
SMB_ERR	2	Индикатор наличия ошибки в данных при приеме пакета 1 – ошибка в принятых данных
	1	Зарезервировано
OR	0	Индикатор переполнения буфера ПРМ при приеме пакета 1 – переполнение буфера

ПРИЛОЖЕНИЕ 4. Алгоритм приема и передачи пакетов

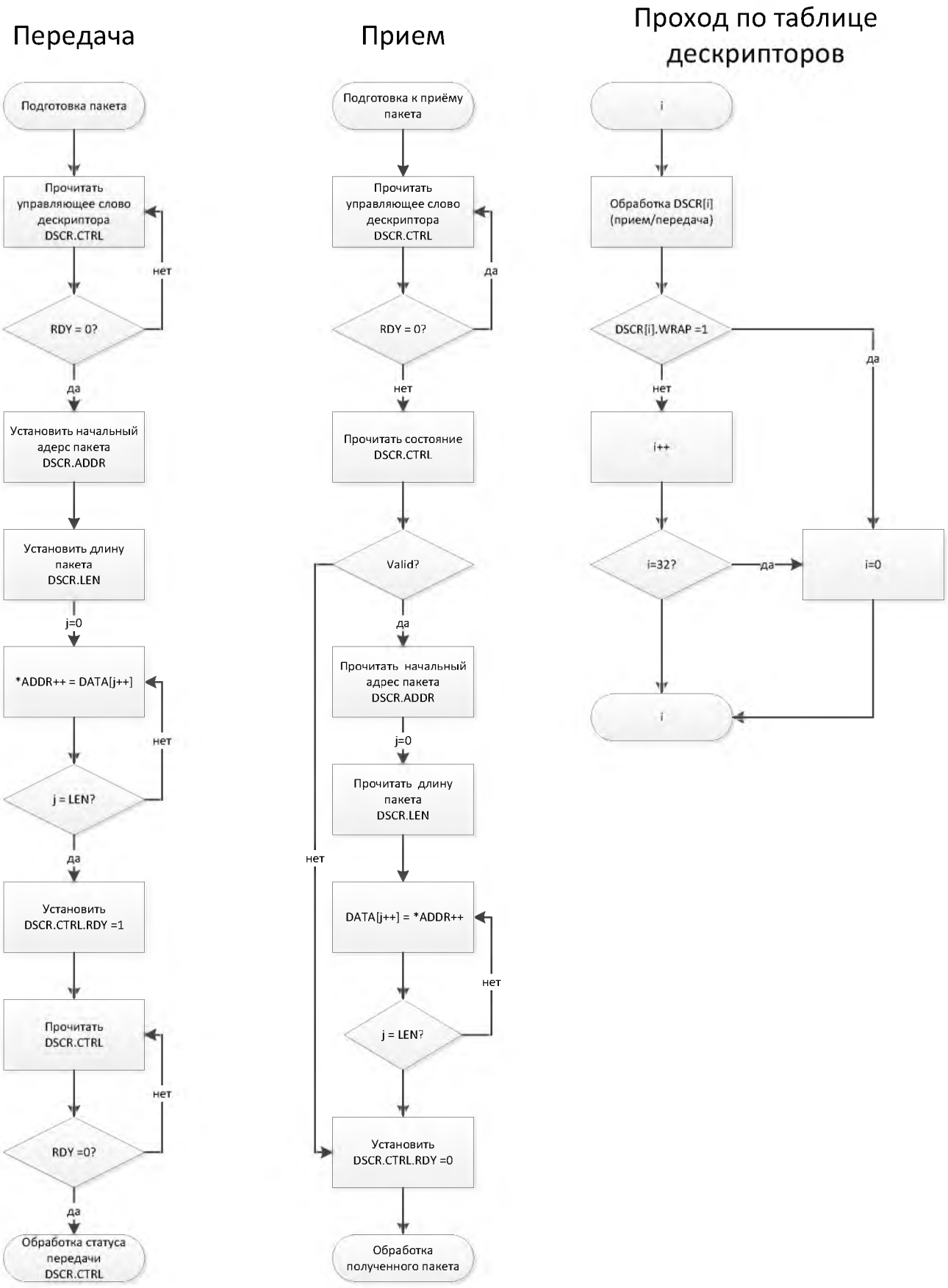


Рисунок 24 – Алгоритм приема и передачи пакетов

ПРИЛОЖЕНИЕ 5. Драйвер (C/C++, TMS320C54x)

Драйвер состоит из четырех файлов (листинг файлов приведен ниже):

«MAC.c»	– содержит определения функций и внутренних переменных;
«BUFF.c»	– содержит объявления буферов и таблиц дескрипторов;
«MAC.h»	– содержит объявления внешних функций и переменных, а также макроопределения констант и регистровой модели контроллера ЛВС;
«MAC_types.h»	– содержит определения структур управления и данных используемых в контроллере ЛВС.

Предоставляемые функции:

MAC_init	– инициализация управляющих структур;
Send_ETH_Pack	– запись пакета в буфер ПРД и соответствующего дескриптора в буфер дескрипторов;
Receive_ETH_Pack	– чтение пакета из буфера ПРМ и очисткой соответствующего дескриптора;
chk_Send_OK	– проверка успешности отправки пакета;
chk_Receive_Ready	– проверка наличия принятого пакета.

Предоставляемые структуры:

основные:

t_MAC	– структура набора регистров управления контроллером ЛВС;
t_TX_Buffer_Descriptor	– структура дескриптора пакета для ПРД;
t_RX_Buffer_Descriptor	– структура дескриптора пакета для ПРМ;
t_ETH_Pack	– структура пакета IEEE802.3/Ethernet;

вспомогательные:

t_MAC_CTRL	– структура полей регистра управления ПРМ/ПРД;
t_MAC_PACKETLEN	– структура указателей минимальной и максимальной длины пакетов;
t_MAC_COLCONF	– структура полей регистра управления обработкой коллизий;
t_INT_SOURCE	– структура полей регистра источников прерываний;
t_INT_MASK	– структура полей регистра маскирования источников прерываний;
t_PHY_CTRL	– структура полей регистра управления модулем PHY;
t_PHY_STAT	– структура полей регистра словосостояния модуля PHY;
t_GCTRL	– структура полей регистра общего управления контроллером ЛВС.

lanc_546.cmd (TMS320C546)

```
/****************************************************************************/
/*                                                                          */
/* LNK.CMD - V2.00  COMMAND FILE FOR LINKING C PROGRAMS                     */
/*                                                                          */
/*      Usage:  lnk500 <obj files...>    -o <out file> -m <map file> lnk.cmd */
/*              cl500  <src files...> -z -o <out file> -m <map file> lnk.cmd */
/*                                                                          */
/*      Description: This file is a sample command file that can be used    */
/*                   for linking programs built with the C54x C Compiler.   */
/*                   This file has been designed to work for                */
/*                   548 C54x device.                                       */
/*                   Use it as a guideline; you may want to make alterations */
/*                   appropriate for the memory layout of the target        */
/*                   system and/or your application.                        */
/*                                                                          */
/*      Notes: (1)   You must specify the directory in which rts.lib is     */
/*                   located.  Either add a "-i<directory>" line to this    */
/*                   file, or use the system environment variable C_DIR to  */
/*                   specify a search path for the libraries.               */
/*                                                                          */
/*             (2)   If the run-time library you are using is not           */
/*                   named rts.lib, be sure to use the correct name here.   */
/****************************************************************************/


MEMORY {
   PAGE 0:                                                      /* program memory */
/*       PROG_RAM (RWX): origin = 0x0080,    length = 0x7f80*/
/*       PROG_EXT (RWX): origin = 0x8000,    length = 0x7f80*/

         PROG_RAM (RWX): origin = 0x1480,    length = 0x2C00
         PROG_EXT (RWX): origin = 0x8000,    length = 0x4000
         VECTORS (RWX):  origin = 0x1400,    length = 0x0080     /* boot interrupt vector table
location */

   PAGE 1:                                                      /* data memory, addresses 0-7Fh are
reserved */
         DATA_RAM (RW):  origin = 0x0080,    length = 0x7f80
         DATA_EXT (RW):  origin = 0x8000,    length = 0x6000
         MAC_RXBF (RW):  origin = 0xE000,    length = 0x0800
         MAC_RXBD (RW):  origin = 0xE800,    length = 0x0100
         MAC_TXBF (RW):  origin = 0xF000,    length = 0x0800
         MAC_TXBD (RW):  origin = 0xF800,    length = 0x0100
         MAC_RG (RW):    origin = 0xFFC0,    length = 0x0020



} /* MEMORY */


SECTIONS {
    .text       >   PROG_RAM | PROG_EXT     PAGE 0  /* code                     */
    .switch     >   PROG_RAM                PAGE 0  /* switch table info        */
    .cinit      >   PROG_RAM                PAGE 0

    .vectors    >   VECTORS                 PAGE 0  /* interrupt vectors         */

    .cio        >   DATA_RAM                PAGE 1  /* C I/O                    */
    .data       >   DATA_RAM | DATA_EXT     PAGE 1  /* initialized data         */
    .bss        >   DATA_RAM | DATA_EXT     PAGE 1  /* global & static variables */
    .const      >   DATA_RAM                PAGE 1  /* constant data            */
    .sysmem     >   DATA_RAM | DATA_EXT     PAGE 1  /* heap                     */
    .stack      >   DATA_RAM | DATA_EXT     PAGE 1  /* stack                    */
    .csldata    >   DATA_RAM                PAGE 1

    .RG_sect    >   MAC_RG                  PAGE 1
    .XB_sect    >   MAC_TXBF                PAGE 1
    .RB_sect    >   MAC_RXBF                PAGE 1
    .XD_sect    >   MAC_TXBD                PAGE 1
    .RD_sect    >   MAC_RXBD                PAGE 1

} /* SECTIONS */
```

MAC.c

```
//---------------------------------------------------------------------------
```

```
// Project: Ethernet MAC
// Author:  Stanislav V. Afanas'ev
// Company: Milandr
// File:    MAC.c
// Version: 1.2
// Date:    22.11.07
//-----------------------------------------------------------------------------
// Description:
//
//-----------------------------------------------------------------------------
#include    "MAC.h"

#define irq_INT0        0x10
#define irq_INT1        0x11
#define irq_INT2        0x12
#define irq_INT3        0x13

// must be at (BASE + 0x1FC0)
extern  t_MAC           MAC_RG;
// must be at (BASE + 0x1800)
extern  t_TX_Buffer_Descriptor  TX_Descriptors[DSCR_CNT];
// must be at (BASE + 0x0800)
extern  t_RX_Buffer_Descriptor  RX_Descriptors[DSCR_CNT];
// must be at (BASE + 0x1000)
extern  unsigned int    TX_buff[BUFF_SIZE];
// must be at (BASE + 0x0000)
extern  unsigned int    RX_buff[BUFF_SIZE];

        t_PACK_state    stat_TX_PKG;
        t_PACK_state    stat_RX_PKG;

        unsigned int    free_size_TX = BUFF_SIZE;

static  unsigned int    *ptr_TX_BUFF;
        t_INT_SOURCE    int_SRC;

        unsigned int    TMP;
        unsigned int    *dst;
        unsigned int    *src;
        unsigned int    err;
        unsigned int    err_count;


interrupt void c_int16()
{
    int_SRC.all = MAC_RG.INT_SOURCE.all;
//  MAC_RG.INT_SOURCE.all = int_SRC.all;
}


//=============================================================================
void    MAC_reset(void)   // done
{
    unsigned int    i;
    MAC_RG.GCTRL.bit.GLBL_RST   = 1;

    RX_Descriptors[0].RX_ctrl.all   = 0x0000;
    TX_Descriptors[0].TX_ctrl.all   = 0x0000;
    ptr_TX_BUFF = (void *) dflt_addr_Tx_BF;

    MAC_MEM_clear();
    MAC_MEM_test();
    MAC_MEM_clear();
}


//=============================================================================
void    MAC_MEM_clear(void)   // done
{
    unsigned int    i;
    for(i = 0; i< sizeof(TX_buff);i++)  TX_buff[i] = 0;
    for(i = 0; i< sizeof(RX_buff);i++)  RX_buff[i] = 0;
}
//=============================================================================
void    MAC_MEM_test(void)   // done
{
    unsigned int    i;
    err = 0;
    err_count = 0;



    for(i = 0; i< sizeof(TX_buff);i++)  TX_buff[i] = 0xFFFF;
    for(i = 0; i< sizeof(RX_buff);i++)  RX_buff[i] = 0xFFFF;
    for(i = 0; i< sizeof(TX_buff);i++)  if(TX_buff[i] != 0xFFFF) err_count = err +1;;
    for(i = 0; i< sizeof(RX_buff);i++)  if(RX_buff[i] != 0xFFFF) err_count = err +1;;


    MAC_MEM_clear();

    //while(1)
```

```
    {
        for(i = 0; i< sizeof(TX_buff);i++)      MAC_MEM_CELL_test(&TX_buff[i], i);
        for(i = 0; i< sizeof(RX_buff);i++)      MAC_MEM_CELL_test(&RX_buff[i], i);
    }

}


//=============================================================================================
void    MAC_MEM_CELL_test(unsigned int *addr, unsigned int value)   // done
{
    *addr = value;
    if(*addr ^ value)   err_count = err +1;//printf("RX_buff init error at %X (%x)\n",addr,*addr);
/*
    *addr ^= value;
    if(*addr ^ value)   err_count = err +1;//printf("RX_buff init error at %X (%x)\n",addr,*addr);

    *addr ^= value;
    if(*addr)           err_count = err +1;//printf("RX_buff init error at %X (%x)\n",addr,*addr);
*/
}


//=============================================================================================
void    MAC_init(void)   // done
{
    MAC_RG.GCTRL.all        = dflt_GCTRL;
    MAC_RG.base_MAC_RG      = dflt_addr_RG;
    MAC_RG.base_MAC_RxBF    = dflt_addr_Rx_BF;
    MAC_RG.base_MAC_TxBF    = dflt_addr_Tx_BF;
    MAC_RG.base_MAC_RxBD    = dflt_addr_Rx_BD;
    MAC_RG.base_MAC_TxBD    = dflt_addr_Tx_BD;

    MAC_RG.MAC_CTRL.all     = dflt_MAC_CTRL;
//BIT_TX_RST | BIT_RX_RST | BIT_READ_CLR_STAT | BIT_BIG_ENDIAN
//BIT_PRO_EN | BIT_BCA_EN | BIT_MCA_EN | BIT_CTRL_FRAME_EN | BIT_LONG_FRAME_EN | BIT_SHORT_FRAME_EN |
BIT_ERR_FRAME_EN
//BIT_HALFD_EN | BIT_BCKOF_DIS | BIT_LB_EN

    MAC_RG.PACKETLEN.MIN_FRAME  = dflt_MinFrame;
    MAC_RG.PACKETLEN.MAX_FRAME  = dflt_MaxFrame;
    MAC_RG.COLLCONF.all         = dflt_CollConfig;
    MAC_RG.IPGT                 = dflt_IPGTx;

    MAC_RG.MAC_ADDR[0]  = dflt_MAC_ADDR_H;
    MAC_RG.MAC_ADDR[1]  = dflt_MAC_ADDR_M;
    MAC_RG.MAC_ADDR[2]  = dflt_MAC_ADDR_T;
    MAC_RG.INT_MASK.all = dflt_INT_MASK;
    MAC_RG.HASH[0]      = 0x0000;
    MAC_RG.HASH[1]      = 0x0000;
    MAC_RG.HASH[2]      = 0x0000;
    MAC_RG.HASH[3]      = 0x8000;

    MAC_RG.PHY_CTRL.all = dflt_PHY_CTRL;
//BIT_PHY_RST | BIT_PHY_EXT_EN | BIT_PHY_ERLY_DV | BIT_PHY_HALFD | BIT_PHY_DLB | BIT_PHY_LB
    MAC_RG.RXBF_HEAD    = dflt_addr_Rx_BF + sizeof_RxBF -1;
    MAC_RG.RXBF_TAIL    = dflt_addr_Rx_BF;


}

//=============================================================================================
int chk_Send_OK(unsigned int *Dscr_Num)
{

    if(!TX_Descriptors[*Dscr_Num].TX_ctrl.bit.RDY)
    {
        stat_TX_PKG.num     = *Dscr_Num;
        stat_TX_PKG.stat    = TX_Descriptors[*Dscr_Num].TX_ctrl.all;
        free_size_TX        +=(TX_Descriptors[*Dscr_Num].len);
        if(free_size_TX > BUFF_SIZE)
            free_size_TX = BUFF_SIZE;
        return  0;
    }
    else
        return  -1;
}

//=============================================================================================
int chk_Receive_Ready(unsigned int *Dscr_Num)
{
    if(!RX_Descriptors[*Dscr_Num].RX_ctrl.bit.EMPTY)
    {
        stat_RX_PKG.num     = *Dscr_Num;
        stat_RX_PKG.stat    = RX_Descriptors[*Dscr_Num].RX_ctrl.all;
        return  0;
    }
    else
        return  -1;
}
```

```
//=============================================================================================
int Send_ETH_Pack(unsigned int *Dscr_Num, t_ETH_Pack *PKG, unsigned int PARAM)
{
    int i;

    if(PKG->len > free_size_TX)
        return  -1;
    if(TX_Descriptors[*Dscr_Num].TX_ctrl.bit.RDY)
        return  -2;


    TX_Descriptors[*Dscr_Num].data  = ptr_TX_BUFF;
    TX_Descriptors[*Dscr_Num].len   = PKG->len +14;

    dst = ptr_TX_BUFF;

    src = (void*) PKG->DA;
    for(i = 3; i >0; i--)
    {
        *dst++ = *src++;
        dst = (void*) (MAC_TX_BUFF_BASE_ADDR | (((unsigned int) dst) & BUFF_SIZE-1));
    }


    src = (void*) PKG->SA;
    for(i = 3; i >0; i--)
    {
        *dst++ = *src++;
        dst = (void*) (MAC_TX_BUFF_BASE_ADDR | (((unsigned int) dst) & BUFF_SIZE-1));
    }


    *dst++ = PKG->len;
    dst = (void*) (MAC_TX_BUFF_BASE_ADDR | (((unsigned int) dst) & BUFF_SIZE-1));

    src = (void*) PKG->data;
    for(i = (PKG->len/2)+1; i >0; i--)
    {
        *dst++ = *src++;
        dst = (void*) (MAC_TX_BUFF_BASE_ADDR | (((unsigned int) dst) & BUFF_SIZE-1));
    }

    ptr_TX_BUFF = dst;
    TX_Descriptors[(*Dscr_Num)++].TX_ctrl.all = PARAM;

    if(PARAM & BIT_TX_DSCR_WRAP)
        *Dscr_Num = 0;
    else
        *Dscr_Num &= ((DSCR_CNT*DSCR_SIZE)-1);
    free_size_TX    -= PKG->len;

    return 0;
}

//=============================================================================================
int Receive_ETH_Pack(unsigned int *Dscr_Num, t_ETH_Pack *PKG, unsigned int PARAM)
{
    int i;
    unsigned int len;

    len = RX_Descriptors[*Dscr_Num].len;

    if(len < 16 )
        return  -1;

    src = (void*) RX_Descriptors[*Dscr_Num].data;
    dst = (void*) PKG->DA;
    for(i = 3; i >0; i--)
    {
        *dst++ = *src++;
        src = (void*) (MAC_RX_BUFF_BASE_ADDR | (((unsigned int) src) & BUFF_SIZE-1));
    }

    dst = (void*) PKG->SA;
    for(i = 3; i >0; i--)
    {
        *dst++ = *src++;
        src = (void*) (MAC_RX_BUFF_BASE_ADDR | (((unsigned int) src) & BUFF_SIZE-1));
    }

    PKG->len = *src++;
    src = (void*) (MAC_RX_BUFF_BASE_ADDR | (((unsigned int) src) & BUFF_SIZE-1));

    dst = (void*) PKG->data;
    for(i = ((PKG->len / 2)+(PKG->len & 0x1)); i >0; i--)
    {
        *dst++ = *src++;
        src = (void*) (MAC_RX_BUFF_BASE_ADDR | (((unsigned int) src) & BUFF_SIZE-1));
    }

    MAC_RG.RXBF_HEAD = (unsigned int) src -1;
```

```
    RX_Descriptors[(*Dscr_Num)++].RX_ctrl.all = PARAM;

    if(PARAM & BIT_RX_DSCR_WRAP)
        *Dscr_Num = 0;
    else
        *Dscr_Num &= ((DSCR_CNT*DSCR_SIZE)-1);


    return 0;
}
```

BUFF.c (TMS320C456)

```
#include    "MAC.h"


#pragma DATA_SECTION(MAC_RG,".RG_sect")
        t_MAC           MAC_RG;
#pragma DATA_SECTION(TX_Descriptors,".XD_sect")
        t_TX_Buffer_Descriptor  TX_Descriptors[DSCR_CNT];
#pragma DATA_SECTION(RX_Descriptors,".RD_sect")
        t_RX_Buffer_Descriptor  RX_Descriptors[DSCR_CNT];
#pragma DATA_SECTION(TX_buff,".XB_sect")
        unsigned int    TX_buff[BUFF_SIZE];
#pragma DATA_SECTION(RX_buff,".RB_sect")
        unsigned int    RX_buff[BUFF_SIZE];
```

MAC.h

```
//-------------------------------------------------------------------------------
// Project: Ethernet MAC
// Author:  Stanislav V. Afanas'ev
// Company: Milandr
// File:    MAC.h
// Version: 1.2
// Date:    22.11.07
//-------------------------------------------------------------------------------
// Description:
//
//-------------------------------------------------------------------------------

//========== MAC address space  ============================================================

#ifndef     __MAC__
#define     __MAC__
#include    "MAC_types.h"

#define     DSCR_CNT                32                  // 32 descriptors
#define     DSCR_SIZE               8                   // 32 descriptors
#define     BUFF_SIZE               (4096/2)            // 4096 kb WORD accessed
#define     BUFF_DSCR_SIZE          (DSCR_CNT*8/2)      // 8-byte length descriptors WORD accessed
#define     RG_SET_SIZE             32                  // 32 2byte registers

#define     MAC_SPC_BASE            0xE000              //  MC/MP Memory space size
#define     MAC_SPC_SIZE            0x2000              //  MC/MP Memory space size

#define     MAC_RX_BUFF_BASE_ADDR   (MAC_SPC_BASE + (0*BUFF_SIZE))              //  RX buffer is
                                                                                placed on top of MC/MP memory space
#define     MAC_TX_BUFF_BASE_ADDR   (MAC_SPC_BASE + (2*BUFF_SIZE))              //  TX buffer is
                                                                                placed on top of MC/MP memory space
                                                                                before RX buffer
#define     MAC_RX_BD_ADDR          (MAC_RX_BUFF_BASE_ADDR + BUFF_SIZE)         //  RX buffer
                                                                                decriptors is placed just before TX
                                                                                buffer
#define     MAC_TX_BD_ADDR          (MAC_TX_BUFF_BASE_ADDR + BUFF_SIZE)         //  TX buffer
                                                                                decriptors is placed just before RX
                                                                                buffer decriptors

#define     MAC_RG_BASE_ADDR        (MAC_SPC_BASE + MAC_SPC_SIZE - 2*RG_SET_SIZE)// MAC register set
                                                                                 is plased on top of MC/MP IO space

//============== INT_MASK/INT_SOURCE ==================
//--------- BIT POSITION -----------------------------
#define     bit_RXF             15  //  Индикатор успешного приёма пакета
#define     bit_RXE             14  //  Индикатор наличия ошибок при приёме пакета
#define     bit_RXC             13  //  Индикатор приёма пакета управления
#define     bit_RXBF_FULL       12  //  Индикатор переполнения буфера ПРМ при приёма пакета
#define     bit_RXD_nREADY      11  //  (зарезервировано)
#define     bit_RXL             10  //  Индикатор приёма пакета длиной более MaxFrame
#define     bit_RXS             9   //  Индикатор приёма пакета длиной менее MinFrame
#define     bit_MISSED_PKG      8   //  (зарезервировано)

#define     bit_TXF             7   //  Индикатор успешной передачи пакета
```

```
#define     bit_TXE             6   //  Индикатор наличия ошибок при передаче пакета
#define     bit_TXC             5   //  Индикатор передачи пакета управления

#define     bit_TXBF_EMPTY      4   // (зарезервировано)
#define     bit_TXBF_EXHAUST    3   // (зарезервировано)
#define     bit_TX_BUSY         0   //  Индикатор принятия и обслуживания пакета Pause


//--------- BIT MASK ----------------------------------
#define     BIT_RXF             (1<<bit_RXF)
#define     BIT_RXE             (1<<bit_RXE)
#define     BIT_RXL             (1<<bit_RXL)
#define     BIT_RXS             (1<<bit_RXS)
#define     BIT_RXC             (1<<bit_RXC)
#define     BIT_RXBF_FULL       (1<<bit_RXBF_FULL)
#define     BIT_RXD_nREADY      (1<<bit_RXD_nREADY)

#define     BIT_TXF             (1<<bit_TXF)
#define     BIT_TXE             (1<<bit_TXE)
#define     BIT_TXC             (1<<bit_TXC)
#define     BIT_TXBF_EMPTY      (1<<bit_TXBF_EMPTY)
#define     BIT_TXBF_EXHAUST    (1<<bit_TXBF_EXHAUST)
#define     BIT_TX_BUSY         (1<<bit_TX_BUSY)

//============= MAC_CNTRL ============================
//--------- BIT POSITION -----------------------------
#define     bit_TX_RST          15      //  Сброс ПРД MAC-уровня (активный уровень "1" )
#define     bit_RX_RST          14      //  Сброс ПРМ MAC-уровня (активный уровень "1" )
#define     bit_TX_DSCR_SCAN_EN 12      //  Переключение ПРД в режим сканирования дескрипторов, когда
                                            переход к следующему осуществляется вне зависимости от
                                            готовности (активный уровень "1" )
#define     bit_PAUSE_EN        11      //  Разрешения обработки Pause Frame (активный уровень "1" )
#define     bit_PRO_EN          10      //  Включение приёма всех пакетов независимо от их MAC-адреса
                                            (активный уровень "1" )
#define     bit_BCA_EN          9       //  Включение приёма всех широковещательных пакетов (активный
                                            уровень "1" )
#define     bit_MCA_EN          8       //  Включение приёма всех пакетов соответствующих HASH-у
                                            (активный уровень "1" )
#define     bit_CTRL_FRAME_EN   7       //  Разрешение приёма управляющих пакетов (активный уровень
                                            "1" )
#define     bit_LONG_FRAME_EN   6       //  Разрешение приёма пакетов длиной более MaxFrame (активный
                                            уровень "1" )
#define     bit_SHORT_FRAME_EN  5       //  Разрешение приёма пакетов длиной менее MinFrame (активный
                                            уровень "1" )
#define     bit_ERR_FRAME_EN    4       //  Разрешение приёма пакетов с ошибками (активный уровень "1"
                                            )
#define     bit_BCKOF_DIS       3       //  Отключение интервала ожидания перед повторением отправки
                                            пакета в случае коллизии (активный уровень "1" )
#define     bit_HALFD_EN        2       //  Переключение в режим полудуплексных приёма-передачи
                                            (активный уровень "1" )
#define     bit_BIG_ENDIAN      1       //  Переключение в режим BIG ENDIAN формата данных (активный
                                            уровень "1" )
#define     bit_LB_EN           0       //  Включение тестового замыкания ПРД на ПРМ м/у уровнями MAC
                                            и PHY (активный уровень "1" )

//--------- BIT MASK ----------------------------------
#define     BIT_TX_RST          (1<<bit_TX_RST)
#define     BIT_RX_RST          (1<<bit_RX_RST)
#define     BIT_TX_DSCR_SCAN_EN (1<<bit_TX_DSCR_SCAN_EN)
#define     BIT_PAUSE_EN        (1<<bit_PAUSE_EN)
#define     BIT_PRO_EN          (1<<bit_PRO_EN)
#define     BIT_BCA_EN          (1<<bit_BCA_EN)
#define     BIT_MCA_EN          (1<<bit_MCA_EN)
#define     BIT_CTRL_FRAME_EN   (1<<bit_CTRL_FRAME_EN)
#define     BIT_LONG_FRAME_EN   (1<<bit_LONG_FRAME_EN)
#define     BIT_SHORT_FRAME_EN  (1<<bit_SHORT_FRAME_EN)
#define     BIT_ERR_FRAME_EN    (1<<bit_ERR_FRAME_EN)
#define     BIT_HALFD_EN        (1<<bit_HALFD_EN)
#define     BIT_BCKOF_DIS       (1<<bit_BCKOF_DIS)
#define     BIT_BIG_ENDIAN      (1<<bit_BIG_ENDIAN)
#define     BIT_LB_EN           (1<<bit_LB_EN)

//============= TX_DESCRIPTOR_CTRL ====================
//--------- BIT POSITION -----------------------------
#define     bit_TX_DSCR_RDY     15      // Индикатор состояния исходящего пакета (1-готов, 0-
                                            отправлен/не заполнен)
#define     bit_TX_DSCR_WRAP    14      // Индикатор последнего дескриптора в таблице (1-переход к
                                            дескриптору #0)
#define     bit_TX_DSCR_IRQ_EN  13      // Разрешение формирования прерываний по передаче (1-вкл., 0-
                                            выкл.)
#define     bit_TX_DSCR_PRE_DIS 10      // Отключение передачи преамбулы (1-передача преамбулы
                                            отключена, 0-стандартный пакет)
#define     bit_TX_DSCR_PAD_DIS 9       // Отключение дополнения пакетов длиной менее MinFrame до
                                            минимальной длины PAD-ми (1-дополнение не производится,
                                            дополнения производится)
#define     bit_TX_DSCR_CRC_DIS 8       // Отключение дополнения пакетов полем CRC32
#define     bit_TX_DSCR_RL      3       // Индикатор исчерпания разрешённого кол-ва повторения в
                                            случае неуспешной отправки исходящего пакета
#define     bit_TX_DSCR_LC      2       // Индикатор наличия Late Collision
#define     bit_TX_DSCR_UR      1       // Underrun
#define     bit_TX_DSCR_CS      0       // Индикатор потери несущей во время передачи пакета
```

```
//--------- BIT MASK ---------------------------------
#define     BIT_TX_DSCR_RDY     (1<<bit_TX_DSCR_RDY)
#define     BIT_TX_DSCR_WRAP    (1<<bit_TX_DSCR_WRAP)
#define     BIT_TX_DSCR_IRQ_EN  (1<<bit_TX_DSCR_IRQ_EN)
#define     BIT_TX_DSCR_PRE_DIS (1<<bit_TX_DSCR_PRE_DIS)
#define     BIT_TX_DSCR_PAD_DIS (1<<bit_TX_DSCR_PAD_DIS)
#define     BIT_TX_DSCR_CRC_DIS (1<<bit_TX_DSCR_CRC_DIS)
#define     BIT_TX_DSCR_RL      (1<<bit_TX_DSCR_RL)
#define     BIT_TX_DSCR_UR      (1<<bit_TX_DSCR_UR)
#define     BIT_TX_DSCR_LC      (1<<bit_TX_DSCR_LC)
       #define     BIT_TX_DSCR_CS      (1<<bit_TX_DSCR_CS)

#define     MSK_TX_DSCR_RTRY    0x00F0      // Retry count

//============= RX_DESCRIPTOR_CTRL ====================
//--------- BIT POSITION -----------------------------
#define     bit_RX_DSCR_RDY     15      // Индикатор состояния исходящего пакета (1-готов к приёму
                                           пакета, 0-пакет принят/не готов)
#define     bit_RX_DSCR_WRAP    14      // Индикатор последнего дескриптора в таблице (1-переход к
                                           дескриптору #0)
#define     bit_RX_DSCR_IRQ_EN  13      // Разрешение формирования прерываний по передаче (1-вкл., 0-
                                           выкл.)
#define     bit_RX_DSCR_nRDY    11      // (зарезервировано)
#define     bit_RX_DSCR_MCA     10      // Индикатор приёма группового пакета с MAC-адресом
                                           соответствующего HASH-таблице
#define     bit_RX_DSCR_BCA     9       // Индикатор приёма широковещательного пакета
#define     bit_RX_DSCR_UCA     8       // Индикатор приёма индивидуального пакета с полным
                                           совпадением MAC-адреса
#define     bit_RX_DSCR_CF      7       // Индикатор приёма пакета управления
#define     bit_RX_DSCR_LF      6       // Индикатор приёма пакета длиной более MaxFrame
#define     bit_RX_DSCR_SF      5       // Индикатор приёма пакета длиной менее MinFrame
#define     bit_RX_DSCR_EF      4       // Индикатор наличия ошибок при приёме пакета (сводный бит по
                                           ошибкам см. ниже)
#define     bit_RX_DSCR_CRC_ERR 3       // Индикатор наличия ошибки CRC при приёме пакета
#define     bit_RX_DSCR_SMB_ERR 2       // Индикатор наличия ошибки в данных при приёме пакета
#define     bit_RX_DSCR_LC      1       // Индикатор наличия Late Collision
#define     bit_RX_DSCR_OR      0       // Индикатор переполнения буфера ПРМ при приёме пакета

//--------- BIT MASK ---------------------------------
#define     BIT_RX_DSCR_RDY     (1<<bit_RX_DSCR_RDY)
#define     BIT_RX_DSCR_WRAP    (1<<bit_RX_DSCR_WRAP)
#define     BIT_RX_DSCR_IRQ_EN  (1<<bit_RX_DSCR_IRQ_EN)
#define     BIT_RX_DSCR_nRDY    (1<<bit_RX_DSCR_nRDY)
#define     BIT_RX_DSCR_SMB_ERR (1<<bit_RX_DSCR_SMB_ERR)
#define     BIT_RX_DSCR_CRC_ERR (1<<bit_RX_DSCR_CRC_ERR)
#define     BIT_RX_DSCR_LC      (1<<bit_RX_DSCR_LC)
#define     BIT_RX_DSCR_OR      (1<<bit_RX_DSCR_OR)
#define     BIT_RX_DSCR_MCA     (1<<bit_RX_DSCR_MCA)
#define     BIT_RX_DSCR_BCA     (1<<bit_RX_DSCR_BCA)
#define     BIT_RX_DSCR_UCA     (1<<bit_RX_DSCR_UCA)
#define     BIT_RX_DSCR_CF      (1<<bit_RX_DSCR_CF)
#define     BIT_RX_DSCR_LF      (1<<bit_RX_DSCR_LF)
#define     BIT_RX_DSCR_SF      (1<<bit_RX_DSCR_SF)
#define     BIT_RX_DSCR_EF      (1<<bit_RX_DSCR_EF)

//=====================================================
#define     bit_DSCR_RDY        bit_TX_DSCR_RDY
#define     BIT_DSCR_RDY        (1<<bit_TX_DSCR_RDY)

//============= COLCONFIG ==============================
//--------- BIT POSITION -----------------------------
#define     MSK_RetriesLimit        0x0F00
#define     MSK_CollisionWindow     0x00FF


//============= PHY_CNTRL ==============================
//--------- BIT POSITION -----------------------------
#define     bit_PHY_RST         15      // Сброс встроенного контроллера PHY-уровня (активный уровень
                                           "1" )
#define     bit_PHY_EXT_EN      14      // Переключение на работу с внешним контроллером PHY-уровня
                                           (активный уровень "1" )
#define     bit_PHY_TXEN        13      // Разрешение работы ПРД встроенного контроллера PHY-уровня
                                           (активный уровень "1" )
#define     bit_PHY_RXEN        12      // Разрешение работы ПРЬ встроенного контроллера PHY-уровня
                                           (активный уровень "1" )
#define     bit_PHY_BASE_2      5       // Переключение на работу с коаксиальным кабелем в режиме
                                           полудуплексных приёма-передачи (1-подлючение по
                                           коаксиальному кабелю,0-подключение по витой паре)
#define     bit_PHY_DIR         4       // Порядок передачи битов в полубайте (1-прямой,0-инверсный)
#define     bit_PHY_EARLY_DV    3       // Включение формирования сигнала RxDV одновременно с сигналом
                                           CRS (активный уровень "1" )
#define     bit_PHY_HALFD       2       // Включение режима полудуплексных приёма-передачи (активный
                                           уровень "1" )
#define     bit_PHY_DLB         1       // Включение тестового замыкания ПРД на ПРМ на входе
                                           контроллера PHY-уровня (активный уровень "1" )
#define     bit_PHY_LB          0       // Включение тестового замыкания ПРД на ПРМ на выходе
                                           контроллера PHY-уровня до аналоговой части ПРМ/ПРД (активный
                                           уровень "1" )

//--------- BIT MASK ---------------------------------
#define     BIT_PHY_RST         (1<<bit_PHY_RST)
#define     BIT_PHY_EXT_EN      (1<<bit_PHY_EXT_EN)
```

```
#define     BIT_PHY_TXEN        (1<<bit_PHY_TXEN)
#define     BIT_PHY_RXEN        (1<<bit_PHY_RXEN)
#define     BIT_PHY_DIR         (1<<bit_PHY_DIR)
#define     BIT_PHY_BASE_2      (1<<bit_PHY_BASE_2)
#define     BIT_PHY_ERLY_DV     (1<<bit_PHY_EARLY_DV)
#define     BIT_PHY_HALFD       (1<<bit_PHY_HALFD)
#define     BIT_PHY_DLB         (1<<bit_PHY_DLB)
#define     BIT_PHY_LB          (1<<bit_PHY_LB)

#define     MSK_PHY_LINK_PERIOD (0x0FC0)

//============= PHY_STAT ==============================
//--------- BIT POSITION -----------------------------
#define     bit_PHY_INT_JAM     13      // Индикатор встроенного контроллера PHY-уровня о передаче JAM
                                           последовательности в случае коллизии
#define     bit_PHY_INT_JAB     12      // Индикатор встроенного контроллера PHY-уровня о превышении
                                           времени передачи максимально разрешённой
#define     bit_PHY_INT_POL     11      // Индикатор встроенного контроллера PHY-уровня о смене
                                           полярности сигналов в линии ПРМ
#define     bit_PHY_INT_LINK    10      // Индикатор встроенного контроллера PHY-уровня о наличии
                                           подключения в линии
#define     bit_PHY_INT_COL     9       // Индикатор встроенного контроллера PHY-уровня о наличии
                                           коллизии в линии
#define     bit_PHY_INT_CRS     8       // Индикатор встроенного контроллера PHY-уровня о наличии
                                           несущей в линии
#define     bit_PHY_EXT_LINK    5       // Индикатор от внешнего контроллера PHY-уровня о наличии
                                           подключения в линии
#define     bit_PHY_EXT_COL     1       // Индикатор от внешнего контроллера PHY-уровня о наличии
                                           коллизии в линии
#define     bit_PHY_EXT_CRS     0       // Индикатор от внешнего контроллера PHY-уровня о наличии
                                           несущей в линии

//--------- BIT MASK ---------------------------------
#define     BIT_PHY_INT_JAM     (1<<bit_PHY_INT_JAM)
#define     BIT_PHY_INT_JAB     (1<<bit_PHY_INT_JAB)
#define     BIT_PHY_INT_POL     (1<<bit_PHY_INT_POL)
#define     BIT_PHY_INT_LINK    (1<<bit_PHY_INT_LINK)
#define     BIT_PHY_INT_COL     (1<<bit_PHY_INT_COL)
#define     BIT_PHY_INT_CRS     (1<<bit_PHY_INT_CRS)
#define     BIT_PHY_EXT_LINK    (1<<bit_PHY_EXT_LINK)
#define     BIT_PHY_EXT_COL     (1<<bit_PHY_EXT_COL)
#define     BIT_PHY_EXT_CRS     (1<<bit_PHY_EXT_CRS)


//============= GCTRL ================================
//  PPI_CTRL
//--------- BIT POSITION -----------------------------
#define     bit_GLBL_RST        15      // Общий сброс всего контроллера (активный уровень "1")
#define     bit_READ_CLR_STAT   14      // Очистка статистики по чтению (активный уровень "1")
#define     bit_SPI_RST         13      // Сброс встроенного контроллера последовательного порта
                                           (активный уровень "1")
#define     bit_RG_inMEM        12      // (зарезервировано)
//--------- BIT MASK ---------------------------------
#define     BIT_GLBL_RST        (1<<bit_GLBL_RST)
#define     BIT_READ_CLR_STAT   (1<<bit_READ_CLR_STAT)
#define     BIT_SPI_RST         (1<<bit_SPI_RST)
#define     BIT_RG_inMEM        (1<<bit_RG_inMEM)

// SPI CTRL
//--------- BIT POSITION -----------------------------
#define     bit_SPI_RX_EDGE     10      // Активный фронт ПРМ контроллера последовательного порта (1-
                                           положительный,0-отрицательный)
#define     bit_SPI_TX_EDGE     9       // Активный фронт ПРД контроллера последовательного порта (1-
                                           положительный,0-отрицательный)
#define     bit_SPI_DIR         8       // Порядок передачи бит (1-MSB,0-LSB)
#define     bit_SPI_FRAME_POL   7       // Активный уровень сигнала кадровой синхронизации
                                           последовательного порта  (1-положительный,0-отрицательный)
#define     bit_SPI_CLK_POL     6       // Полярность тактового сигнала последовательного порта  (1-
                                           инверсная,0-прямая)
#define     bit_SPI_CLK_PHASE   5       // Фаза тактового сигнала последовательного порта  (1-
                                           инверсная,0-прямая)
#define     bit_SPI_MASTER      4       // (зарезервировано)
//--------- BIT MASK ---------------------------------
#define     BIT_SPI_RX_EDGE     (1<<bit_SPI_RX_EDGE)
#define     BIT_SPI_TX_EDGE     (1<<bit_SPI_TX_EDGE)
#define     BIT_SPI_DIR         (1<<bit_SPI_DIR)
#define     BIT_SPI_FPOL        (1<<bit_SPI_FRAME_POL)
#define     BIT_SPI_CPOL        (1<<bit_SPI_CLK_POL)
#define     BIT_SPI_CPHASE      (1<<bit_SPI_CLK_PHASE)
#define     BIT_SPI_MASTER      (1<<bit_SPI_MASTER)
#define     MSK_SPI_DIV         (0x0007)

//============= DEFAULT VALUES ======================
#define     sizeof_RxBF         2048
#define     sizeof_TxBF         2048
#define     sizeof_RxBD         32*4
#define     sizeof_TxBD         32*4
#define     sizeof_RG           32                  // 32 regs 2 bytes each

#define     depth_RxBF          11                  // = log2(size_BD)
#define     depth_TxBF          11                  // = log2(size_BD)
#define     depth_RxBD          7                   // = log2(size_BD)
```

```
#define     depth_TxBD          7                   // = log2(size_BD)
#define     depth_RG            5                   // = log2(size_BD)

#define     mask_RxBF           (sizeof_RxBF -1)    // 0x07FF
#define     mask_TxBF           (sizeof_TxBF -1)    // 0x07FF
#define     mask_RxBD           (sizeof_RxBD -1)    // 0x007F
#define     mask_TxBD           (sizeof_TxBD -1)    // 0x007F
#define     mask_RG             (sizeof_RG -1)      // 0x001F


#define     dflt_SPI_DIV        (0x2 & MSK_SPI_DIV)
#define     dflt_SPI_CTRL       (BIT_SPI_FPOL | BIT_SPI_DIR | dflt_SPI_DIV  | BIT_SPI_TX_EDGE)
#define     dflt_PPI_CTRL       (BIT_READ_CLR_STAT | BIT_RG_inMEM)

#define     dflt_GCTRL          (dflt_PPI_CTRL | dflt_SPI_CTRL)
#define     dflt_addr_RG        (MAC_SPC_BASE + MAC_SPC_SIZE -2*sizeof_RG)
#define     dflt_addr_Rx_BF     (MAC_SPC_BASE + 0*sizeof_RxBF)
#define     dflt_addr_Tx_BF     (MAC_SPC_BASE + 2*sizeof_TxBF)
#define     dflt_addr_Rx_BD     (dflt_addr_Rx_BF + sizeof_RxBF)
#define     dflt_addr_Tx_BD     (dflt_addr_Tx_BF + sizeof_TxBF)


#define     dflt_MAC_CTRL       (BIT_TX_RST | BIT_RX_RST | BIT_CTRL_FRAME_EN | BIT_SHORT_FRAME_EN)
//BIT_TX_RST | BIT_RX_RST | BIT_READ_CLR_STAT | BIT_BIG_ENDIAN
//BIT_PRO_EN | BIT_BCA_EN | BIT_MCA_EN | BIT_CTRL_FRAME_EN | BIT_LONG_FRAME_EN | BIT_SHORT_FRAME_EN |
BIT_ERR_FRAME_EN
//BIT_HALFD_EN | BIT_BCKOF_DIS | BIT_LB_EN

#define     dflt_MinFrame       0x0040      //  64 bytes
#define     dflt_MaxFrame       0x0600      //  1500 bytes

#define     dflt_MAC_ADDR_H     0x0123      // MAC: 01_23_45_67_89_AB
#define     dflt_MAC_ADDR_M     0x4567
#define     dflt_MAC_ADDR_T     0x89ab

#define     dflt_CollConfig     0x0F40
#define     dflt_IPGTx          0x0060
#define     dflt_INT_MASK       0x0000      // All interrupts disabled

#define     dflt_HASH0          0x0000      // HAH is empty
#define     dflt_HASH1          0x0000
#define     dflt_HASH2          0x0000
#define     dflt_HASH3          0x8000


#define     dflt_PHY_LINK_PERIOD    (0x0B<<6)
#define     dflt_PHY_CTRL           (BIT_PHY_RST | BIT_PHY_TXEN | BIT_PHY_RXEN | BIT_PHY_DIR |
dflt_PHY_LINK_PERIOD)
//BIT_PHY_RST | BIT_PHY_EXT_EN | BIT_PHY_ERLY_DV | BIT_PHY_HALFD | BIT_PHY_DLB | BIT_PHY_LB

// ==================== SPI PROTOCOL defines ==========================================
//SPI SubCMD bits
#define bit_BigEndian           0
#define bit_inMEM               1
#define bit_size0               2
#define bit_size1               3

#define BIG_ENDN                (1<<bit_BigEndian)
#define MEM_WRITE               (1<<bit_inMEM)

#define SIZE8                   0x00
#define SIZE16                  0x04
#define SIZE24                  0x08
#define SIZE32                  0x0c


#define SPI_SOP                 0xDB
#define SPI_dSOP                0xDD
#define SPI_DSOP                0xDD
#define SPI_FILL                0x00
//#define   SPI_FILL                SPI_SOP

#define cmd_RESET               0x0F
#define cmd_SelectSPI           0x01
#define cmd_SelectPPI           0x02
#define cmd_Write               0x03
#define cmd_Read                0x04

#define SPI_CMD_RESET           (cmd_RESET     <<4)
#define SPI_CMD_SelectSPI       (cmd_SelectSPI <<4)
#define SPI_CMD_SelectPPI       (cmd_SelectPPI <<4)
#define SPI_CMD_WRITE           (cmd_Write     <<4)
#define SPI_CMD_READ            (cmd_Read      <<4)


// SPI Errors
#define err_NoErr               0x00
#define err_ErrCMD              0x01
#define err_DblSOPinData        0x02
#define err_SnglSOPinData       0x03
#define err_TransferAbort       0x04
```

```
// ================== FUNCTION DECLARATIONS ==========================================
void    MAC_reset(void);
void    MAC_MEM_clear(void);
void    MAC_MEM_test(void);
void    MAC_MEM_CELL_test(unsigned int *addr, unsigned int value);
void    MAC_init(void);
int     Send_ETH_Pack(unsigned int *Dscr_Num, t_ETH_Pack *PKG, unsigned int PARAM);
int     Receive_ETH_Pack (unsigned int *Dscr_Num, t_ETH_Pack *PKG, unsigned int PARAM);
int     chk_Send_OK (unsigned int *Dscr_Num);
int     chk_Receive_Ready (unsigned int *Dscr_Num);


#endif  // __MAC__
```

MAC_types.h

```
//-------------------------------------------------------------------------------
// Project: Ethernet MAC
// Author:  Stanislav V. Afanas'ev
// Company: Milandr
// File:    MAC_types.h
// Version: 1.2
// Date:    22.11.07
//-------------------------------------------------------------------------------
// Description:
//
//-------------------------------------------------------------------------------
#ifndef     __MAC_TYPES__
#define     __MAC_TYPES__

#ifdef      __TMS320C50__
#define     __BIT_ORDER_REVERSE__
#endif      //__TMS320C50__

typedef struct  PACK_state
{
            unsigned int    num;            // Номер дескриптора
            unsigned int    stat;           // Слово-состояние дескриптора
} t_PACK_state;


typedef struct  TX_BD_ctrl
{
#ifdef  __BIT_ORDER_REVERSE__
    volatile unsigned   CS          : 1;    // Индикатор потери несущей во время передачи пакета
    volatile unsigned   UR          : 1;    // Индикатор наличия Late Collision
    volatile unsigned   LC          : 1;    // Underrun
    volatile unsigned   RL          : 1;    // Индикатор исчерпания разрешённого кол-ва повторения в
                                               случае неуспешной отправки исходящего пакета
    volatile unsigned   RTRY        : 4;    // Счетчик кол-ва попыток передачи исходящего пакета
             unsigned   CRC_DIS     : 1;    // Отключение дополнения пакетов полем CRC32
             unsigned   PAD_DIS     : 1;    // Отключение дополнения пакетов длиной менее MinFrame до
                                               минимальной длины PAD-ми (1-дополнение не производится,
                                               дополнения производится)
             unsigned   PRE_DIS     : 1;    // Отключение передачи преамбулы (1-передача преамбулы
                                               отключена, 0-стандартный пакет)
             unsigned   reserved    : 2;    // (зарезервировано)
             unsigned   IRQ_EN      : 1;    // Разрешение формирования прерываний по передаче (1-вкл.,
                                               0-выкл.)
             unsigned   WRAP        : 1;    // Индикатор последнего дескриптора в таблице (1-переход к
                                               дескриптору #0)
    volatile unsigned   RDY         : 1;    // Индикатор состояния исходящего пакета (1-готов, 0-
                                               отправлен/не заполнен)
#else   //__BIT_ORDER_REVERSE__
    volatile unsigned   RDY         : 1;    // Индикатор состояния исходящего пакета (1-готов, 0-
                                               отправлен/не заполнен)
             unsigned   WRAP        : 1;    // Индикатор последнего дескриптора в таблице (1-переход к
                                               дескриптору #0)
             unsigned   IRQ_EN      : 1;    // Разрешение формирования прерываний по передаче (1-вкл.,
                                               0-выкл.)
             unsigned   reserved    : 2;    // (зарезервировано)
             unsigned   PRE_DIS     : 1;    // Отключение передачи преамбулы (1-передача преамбулы
                                               отключена, 0-стандартный пакет)
             unsigned   PAD_DIS     : 1;    // Отключение дополнения пакетов длиной менее MinFrame до
                                               минимальной длины PAD-ми (1-дополнение не производится,
                                               дополнения производится)
             unsigned   CRC_DIS     : 1;    // Отключение дополнения пакетов полем CRC32
    volatile unsigned   RTRY        : 4;    // Счетчик кол-ва попыток передачи исходящего пакета
    volatile unsigned   RL          : 1;    // Индикатор исчерпания разрешённого кол-ва повторения в
                                               случае неуспешной отправки исходящего пакета
    volatile unsigned   LC          : 1;    // (зарезервировано)
    volatile unsigned   UR          : 1;    // Индикатор наличия Late Collision
    volatile unsigned   CS          : 1;    // Индикатор потери несущей во время передачи пакета
#endif  //__BIT_ORDER_REVERSE__
} t_TX_BD_ctrl;
```

```
typedef struct  RX_BD_ctrl
{
#ifdef  __BIT_ORDER_REVERSE__
    volatile unsigned   OR          : 1;    // Индикатор переполнения буфера ПРМ при приёме пакета
    volatile unsigned   LC          : 1;    // Индикатор наличия Late Collision
    volatile unsigned   SMB_ERR     : 1;    // Индикатор наличия ошибки в данных при приёме пакета
    volatile unsigned   CRC_ERR     : 1;    // Индикатор наличия ошибки CRC при приёме пакета
    volatile unsigned   EF          : 1;    // Индикатор наличия ошибок при приёме пакета (сводный бит
                                               по ошибкам см. ниже)
    volatile unsigned   SF          : 1;    // Индикатор приёма пакета длиной менее MinFrame
    volatile unsigned   LF          : 1;    // Индикатор приёма пакета длиной более MaxFrame
    volatile unsigned   CF          : 1;    // Индикатор приёма пакета управления
    volatile unsigned   UCA         : 1;    // Индикатор приёма индивидуального пакета с полным
                                               совпадением MAC-адреса
    volatile unsigned   MCA         : 1;    // Индикатор приёма широковещательного пакета
    volatile unsigned   BCA         : 1;    // Индикатор приёма группового пакета с MAC-адресом
                                               соответствующего HASH-таблице
             unsigned   reserved    : 2;    // (зарезервировано)
             unsigned   IRQ         : 1;    // Разрешение формирования прерываний по передаче (1-вкл.,
                                               0-выкл.)
             unsigned   WRAP        : 1;    // Индикатор последнего дескриптора в таблице (1-переход к
                                               дескриптору #0)
    volatile unsigned   EMPTY       : 1;    // Индикатор состояния исходящего пакета (1-готов к приёму
                                               пакета, 0-пакет принят/не готов)
#else   //__BIT_ORDER_REVERSE__
    volatile unsigned   EMPTY       : 1;    // Индикатор состояния исходящего пакета (1-готов к приёму
                                               пакета, 0-пакет принят/не готов)
             unsigned   WRAP        : 1;    // Индикатор последнего дескриптора в таблице (1-переход к
                                               дескриптору #0)
             unsigned   IRQ         : 1;    // Разрешение формирования прерываний по передаче (1-вкл.,
                                               0-выкл.)
             unsigned   reserved    : 2;    // (зарезервировано)
    volatile unsigned   BCA         : 1;    // Индикатор приёма группового пакета с MAC-адресом
                                               соответствующего HASH-таблице
    volatile unsigned   MCA         : 1;    // Индикатор приёма широковещательного пакета
    volatile unsigned   UCA         : 1;    // Индикатор приёма индивидуального пакета с полным
                                               совпадением MAC-адреса
    volatile unsigned   CF          : 1;    // Индикатор приёма пакета управления
    volatile unsigned   LF          : 1;    // Индикатор приёма пакета длиной более MaxFrame
    volatile unsigned   SF          : 1;    // Индикатор приёма пакета длиной менее MinFrame
    volatile unsigned   EF          : 1;    // Индикатор наличия ошибок при приёме пакета (сводный бит
                                               по ошибкам см. ниже)
    volatile unsigned   CRC_ERR     : 1;    // Индикатор наличия ошибки CRC при приёме пакета
    volatile unsigned   SMB_ERR     : 1;    // Индикатор наличия ошибки в данных при приёме пакета
    volatile unsigned   LC          : 1;    // Индикатор наличия Late Collision
    volatile unsigned   OR          : 1;    // Индикатор переполнения буфера ПРМ при приёме пакета
#endif  //__BIT_ORDER_REVERSE__
} t_RX_BD_ctrl;

typedef union   TX_ctrl
{
            unsigned int    all;
            t_TX_BD_ctrl    bit;
} t_TX_ctrl;

typedef union RX_ctrl
{
            unsigned int    all;
            t_RX_BD_ctrl    bit;
} t_RX_ctrl;

typedef struct  TX_Buffer_Descriptor
{
            t_TX_ctrl       TX_ctrl;            // Слово-состояние отправки пакета
            unsigned int    len;                // Полная длина пакета (включая поля SA, DA и Length)
                                                    !!! в байтах !!!
            unsigned int    NULL;               // Старшая часть указателя данных пакета (0x0000)
            unsigned int    *data;              // Указатель на данные пакета (включая поля SA, DA и
                                                    Length)
} t_TX_Buffer_Descriptor;

typedef struct  RX_Buffer_Descriptor
{
            t_RX_ctrl       RX_ctrl;            // Слово-состояние приёма пакета
            unsigned int    len;                // Полная длина пакета (включая поля SA, DA и Length)
                                                    !!! в байтах !!!
            unsigned int    NULL;               // Старшая часть указателя данных пакета (0x0000)
            unsigned int    *data;              // Указатель на данные пакета (включая поля SA, DA и
                                                    Length)
} t_RX_Buffer_Descriptor;


typedef struct  ETH_Packet
{
            unsigned int    *DA;                // Указатель на адрес получателя пакета (DA)
            unsigned int    *SA;                // Указатель на адрес отправителя пакета (SA)
            unsigned int    len;                // Длина пакета
            unsigned int    *data;              // Указатель на данные пакета
} t_ETH_Pack;
```

```
typedef struct  bits_MAC_CTRL
{
#ifdef  __BIT_ORDER_REVERSE__
            unsigned    LOOPBACK_EN     : 1;    // Включение тестового замыкания ПРД на ПРМ м/у
                                                    уровнями MAC и PHY (активный уровень "1" )
            unsigned    BIG_ENDIAN      : 1;    // Переключение в режим BIG ENDIAN формата данных
                                                    (активный уровень "1" )
            unsigned    HALFD_EN        : 1;    // Переключение в режим полудуплексных приёма-передачи
                                                    (активный уровень "1" )
            unsigned    BCKOF_DIS       : 1;    // Отключение интервала ожидания перед повторением
                                                    отправки пакета в случае коллизии (активный уровень
                                                    "1" )
            unsigned    ERR_FRAME_EN    : 1;    // Разрешение приёма пакетов с ошибками (активный
                                                    уровень "1" )
            unsigned    SHORT_FRAME_EN  : 1;    // Разрешение приёма пакетов длиной менее MinFrame
                                                    (активный уровень "1" )
            unsigned    LONG_FRAME_EN   : 1;    // Разрешение приёма пакетов длиной более MaxFrame
                                                    (активный уровень "1" )
            unsigned    CTRL_FRAME_EN   : 1;    // Разрешение приёма управляющих пакетов (активный
                                                    уровень "1" )
            unsigned    MCA_EN          : 1;    // Включение приёма всех пакетов соответствующих HASH-
                                                    у (активный уровень "1" )
            unsigned    BCA_EN          : 1;    // Включение приёма всех широковещательных пакетов
                                                    (активный уровень "1" )
            unsigned    PRO_EN          : 1;    // Включение приёма всех пакетов независимо от их MAC-
                                                    адреса (активный уровень "1" )
            unsigned    PAUSE_EN        : 1;    // Разрешения обработки Pause Frame (активный уровень
                                                    "1" )
            unsigned    reserved        : 2;    // Переключение ПРД в режим сканирования дескрипторов,
                                                    когда переход к следующему осуществляется вне
                                                    зависимости от готовности (активный уровень "1" )
            unsigned    RX_RST          : 1;    // Сброс ПРМ MAC-уровня (активный уровень "1" )
            unsigned    TX_RST          : 1;    // Сброс ПРД MAC-уровня (активный уровень "1" )
#else   //__BIT_ORDER_REVERSE__
            unsigned    TX_RST          : 1;    // Сброс ПРД MAC-уровня (активный уровень "1" )
            unsigned    RX_RST          : 1;    // Сброс ПРМ MAC-уровня (активный уровень "1" )
            unsigned    reserved        : 2;    // Переключение ПРД в режим сканирования дескрипторов,
                                                    когда переход к следующему осуществляется вне
                                                    зависимости от готовности (активный уровень "1" )
            unsigned    PAUSE_EN        : 1;    // Разрешения обработки Pause Frame (активный уровень
                                                    "1" )
            unsigned    PRO_EN          : 1;    // Включение приёма всех пакетов независимо от их MAC-
                                                    адреса (активный уровень "1" )
            unsigned    BCA_EN          : 1;    // Включение приёма всех широковещательных пакетов
                                                    (активный уровень "1" )
            unsigned    MCA_EN          : 1;    // Включение приёма всех пакетов соответствующих HASH-
                                                    у (активный уровень "1" )
            unsigned    CTRL_FRAME_EN   : 1;    // Разрешение приёма управляющих пакетов (активный
                                                    уровень "1" )
            unsigned    LONG_FRAME_EN   : 1;    // Разрешение приёма пакетов длиной более MaxFrame
                                                    (активный уровень "1" )
            unsigned    SHORT_FRAME_EN  : 1;    // Разрешение приёма пакетов длиной менее MinFrame
                                                    (активный уровень "1" )
            unsigned    ERR_FRAME_EN    : 1;    // Разрешение приёма пакетов с ошибками (активный
                                                    уровень "1" )
            unsigned    BCKOF_DIS       : 1;    // Отключение интервала ожидания перед повторением
                                                    отправки пакета в случае коллизии (активный уровень
                                                    "1" )
            unsigned    HALFD_EN        : 1;    // Переключение в режим полудуплексных приёма-передачи
                                                    (активный уровень "1" )
            unsigned    BIG_ENDIAN      : 1;    // Переключение в режим BIG ENDIAN формата данных
                                                    (активный уровень "1" )
            unsigned    LOOPBACK_EN     : 1;    // Включение тестового замыкания ПРД на ПРМ м/у
                                                    уровнями MAC и PHY (активный уровень "1" )
#endif  //__BIT_ORDER_REVERSE__
} t_bits_MAC_CTRL;

typedef struct  bits_MAC_INT
{
#ifdef  __BIT_ORDER_REVERSE__
    volatile unsigned   TX_BUSY         : 1;    // Индикатор принятия и обслуживания пакета Pause
    volatile unsigned   reserved        : 4;    // (зарезервировано)
    volatile unsigned   TXC             : 1;    // Индикатор передачи пакета управления
    volatile unsigned   TXE             : 1;    // Индикатор наличия ошибок при передаче пакета
    volatile unsigned   TXF             : 1;    // Индикатор успешной передачи пакета

    volatile unsigned   Missed_PKG      : 1;    // (зарезервировано)
    volatile unsigned   RXS             : 1;    // Индикатор приёма пакета длиной менее MinFrame
    volatile unsigned   RXL             : 1;    // Индикатор приёма пакета длиной более MaxFrame
    volatile unsigned   RXBD_nREADY     : 1;    // (зарезервировано)
    volatile unsigned   RXBF_FULL       : 1;    // Индикатор переполнения буфера ПРМ при приёма пакета
    volatile unsigned   RXC             : 1;    // Индикатор приёма пакета управления
    volatile unsigned   RXE             : 1;    // Индикатор наличия ошибок при приёме пакета
    volatile unsigned   RXF             : 1;    // Индикатор успешного приёма пакета
#else   //__BIT_ORDER_REVERSE__
    volatile unsigned   RXF             : 1;    // Индикатор успешного приёма пакета
    volatile unsigned   RXE             : 1;    // Индикатор наличия ошибок при приёме пакета
    volatile unsigned   RXC             : 1;    // Индикатор приёма пакета управления
    volatile unsigned   RXBF_FULL       : 1;    // Индикатор переполнения буфера ПРМ при приёма пакета
    volatile unsigned   RXBD_nREADY     : 1;    // (зарезервировано)
    volatile unsigned   RXL             : 1;    // Индикатор приёма пакета длиной более MaxFrame
    volatile unsigned   RXS             : 1;    // Индикатор приёма пакета длиной менее MinFrame
```

```
    volatile unsigned   Missed_PKG      : 1;    // (зарезервировано)

    volatile unsigned   TXF             : 1;    // Индикатор успешной передачи пакета
    volatile unsigned   TXE             : 1;    // Индикатор наличия ошибок при передаче пакета
    volatile unsigned   TXC             : 1;    // Индикатор передачи пакета управления
    volatile unsigned   reserved        : 4;    // (зарезервировано)
    volatile unsigned   TX_BUSY         : 1;    // Индикатор принятия и обслуживания пакета Pause
#endif  //__BIT_ORDER_REVERSE__
} t_bits_MAC_INT;
/*
typedef struct  bits_MAC_INT_MASK
{
#ifdef  __BIT_ORDER_REVERSE__
    unsigned    TX_BUSY     : 1;    //  Busy
    unsigned                : 4;    //  reserved
    unsigned    TXC         : 1;    //  Transmit Error
    unsigned    TXE         : 1;    //  Receive Frame
    unsigned    TXF         : 1;    //  Receive Error

    unsigned                : 3;    //  reserved
    unsigned    RXBD_nREADY : 1;    //  Receive Frame
    unsigned    RXBF_FULL   : 1;    //  Receive Error
    unsigned    BXC         : 1;    //  Busy
    unsigned    RXE         : 1;    //  Transmit Control Frame
    unsigned    RXF         : 1;    //  Receive Control Frame
#else   //__BIT_ORDER_REVERSE__
#endif  //__BIT_ORDER_REVERSE__
} t_bits_MAC_INT_MASK;
*/
typedef struct  MAC_PACKETLEN
{
            unsigned int        MIN_FRAME;      // Минимальная допустимая длина пакета
            unsigned int        MAX_FRAME;      // Максимальная допустимая длина пакета
} t_MAC_PACKETLEN;

typedef struct  bits_MAC_COLLCONF
{
#ifdef  __BIT_ORDER_REVERSE__
            unsigned    COLLVALID       : 8;    // Допустимое время появления коллизии в линии (в 4xBT
= 400нс)
            unsigned    MAXRET          : 4;    // Максимальное кол-во повторений отправки пакета
            unsigned    reserved        : 4;    // (зарезервировано)
#else   //__BIT_ORDER_REVERSE__
            unsigned    reserved        : 4;    // (зарезервировано)
            unsigned    MAXRET          : 4;    // Максимальное кол-во повторений отправки пакета
            unsigned    COLLVALID       : 8;    // Допустимое время появления коллизии в линии (в 4xBT
= 400нс)
#endif  //__BIT_ORDER_REVERSE__
} t_bits_MAC_COLLCONF;


typedef struct  bits_GCTRL
{
#ifdef  __BIT_ORDER_REVERSE__
            unsigned    GLBL_RST        :1;     // Общий сброс всего контроллера (активный уровень
                                                  "1")
            unsigned    READ_CLR_STAT   :1;     // Очистка статистики по чтению (активный уровень "1")
            unsigned    SPI_RST         :1;     // Сброс встроенного контроллера последовательного
                                                  порта (активный уровень "1")
            unsigned    reserved        :2;     // (зарезервировано)
            unsigned    SPI_RX_EDGE     :1;     // Активный фронт ПРМ контроллера последовательного
                                                  порта (1-положительный,0-отрицательный)
            unsigned    SPI_TX_EDGE     :1;     // Активный фронт ПРД контроллера последовательного
                                                  порта (1-положительный,0-отрицательный)
            unsigned    SPI_DIR         :1;     // Порядок передачи бит (1-MSB,0-LSB)
            unsigned    SPI_FRAME_POL   :1;     // Активный уровень сигнала кадровой синхронизации
                                                  последовательного порта  (1-положительный,0-
                                                  отрицательный)
            unsigned    SPI_CLK_POL     :1;     // Полярность тактового сигнала последовательного
                                                  порта  (1-инверсная,0-прямая)
            unsigned    SPI_CLK_PHASE   :1;     // Фаза тактового сигнала последовательного порта  (1-
                                                  инверсная,0-прямая)
            unsigned    SPI_MASTER      :1;     // (зарезервировано)
            unsigned    SPI_DIV         :4;     // (зарезервировано)
#else   //__BIT_ORDER_REVERSE__
            unsigned    SPI_DIV         :4;     // (зарезервировано)
            unsigned    SPI_MASTER      :1;     // (зарезервировано)
            unsigned    SPI_CLK_PHASE   :1;     // Фаза тактового сигнала последовательного порта  (1-
                                                  инверсная,0-прямая)
            unsigned    SPI_CLK_POL     :1;     // Полярность тактового сигнала последовательного
                                                  порта  (1-инверсная,0-прямая)
            unsigned    SPI_FRAME_POL   :1;     // Активный уровень сигнала кадровой синхронизации
                                                  последовательного порта  (1-положительный,0-
                                                  отрицательный)
            unsigned    SPI_DIR         :1;     // Порядок передачи бит (1-MSB,0-LSB)
            unsigned    SPI_TX_EDGE     :1;     // Активный фронт ПРД контроллера последовательного
                                                  порта (1-положительный,0-отрицательный)
            unsigned    SPI_RX_EDGE     :1;     // Активный фронт ПРМ контроллера последовательного
                                                  порта (1-положительный,0-отрицательный)
            unsigned    reserved        :2;     // (зарезервировано)
            unsigned    SPI_RST         :1;     // Сброс встроенного контроллера последовательного
                                                  порта (активный уровень "1")
```

```
            unsigned    READ_CLR_STAT   :1;     // Очистка статистики по чтению (активный уровень "1")
            unsigned    GLBL_RST        :1;     // Общий сброс всего контроллера (активный уровень
                                                   "1")
#endif  //__BIT_ORDER_REVERSE__
} t_bits_GCTRL;


typedef struct  bits_PHY_CTRL
{
#ifdef  __BIT_ORDER_REVERSE__
            unsigned    PHY_LB          :1;     // Включение тестового замыкания ПРД на ПРМ на выходе
                                                   контроллера PHY-уровня до аналоговой части ПРМ/ПРД
                                                   (активный уровень "1" )
            unsigned    PHY_DLB         :1;     // Включение тестового замыкания ПРД на ПРМ на входе
                                                   контроллера PHY-уровня (активный уровень "1" )
            unsigned    PHY_HALFD       :1;     // Включение режима полудуплексных приёма/передачи
                                                   (активный уровень "1" )
            unsigned    PHY_EARLY_DV    :1;     // Включение формирования сигнала RxDV одновременно с
                                                   сигналом CRS (активный уровень "1" )
            unsigned    PHY_DIR         :1;     // Порядок передачи битов в полубайте (1-прямой,0-
                                                   инверсный)
            unsigned    PHY_BASE_2      :1;     // Переключение на работу с коаксиальным кабелем в
                                                   режиме полудуплексных приёма-передачи (1-подлючение
                                                   по коаксиальному кабелю,0-подключение по витой
                                                   паре)
            unsigned    PHY_LINK_PERIOD :6;     // Период следования LINK импульсов -1
                                                   (диапазон 1..64 мс)
                                                // Период ожидания LINK импульсов вдвое больше
                                                   заданного периода следования LINK импульсов
            unsigned    PHY_RXEN        :1;     // Разрешение работы ПРЬ встроенного контроллера PHY-
                                                   уровня (активный уровень "1" )
            unsigned    PHY_TXEN        :1;     // Разрешение работы ПРД встроенного контроллера PHY-
                                                   уровня (активный уровень "1" )
            unsigned    PHY_EXT_EN      :1;     // Переключение на работу с внешним контроллером PHY-
                                                   уровня (активный уровень "1" )
            unsigned    PHY_RST         :1;     // Сброс встроенного контроллера PHY-уровня (активный
                                                   уровень "1" )
#else   //__BIT_ORDER_REVERSE__
            unsigned    PHY_RST         :1;     // Сброс встроенного контроллера PHY-уровня (активный
                                                   уровень "1" )
            unsigned    PHY_EXT_EN      :1;     // Переключение на работу с внешним контроллером PHY-
                                                   уровня (активный уровень "1" )
            unsigned    PHY_TXEN        :1;     // Разрешение работы ПРД встроенного контроллера PHY-
                                                   уровня (активный уровень "1" )
            unsigned    PHY_RXEN        :1;     // Разрешение работы ПРЬ встроенного контроллера PHY-
                                                   уровня (активный уровень "1" )
            unsigned    PHY_LINK_PERIOD :6;     // Период следования LINK импульсов -1 (диапазон 1..64
                                                   мс)
                                                // Период ожидания LINK импульсов вдвое больше
                                                   заданного периода следования LINK импульсов
            unsigned    PHY_BASE_2      :1;     // Переключение на работу с коаксиальным кабелем в
                                                   режиме полудуплексных приёма-передачи (1-подлючение
                                                   по коаксиальному кабелю,0-подключение по витой
                                                   паре)
            unsigned    PHY_DIR         :1;     // Порядок передачи битов в полубайте (1-прямой,0-
                                                   инверсный)
            unsigned    PHY_EARLY_DV    :1;     // Включение формирования сигнала RxDV одновременно с
                                                   сигналом CRS (активный уровень "1" )
            unsigned    PHY_HALFD       :1;     // Включение режима полудуплексных приёма-передачи
                                                   (активный уровень "1" )
            unsigned    PHY_DLB         :1;     // Включение тестового замыкания ПРД на ПРМ на входе
                                                   контроллера PHY-уровня (активный уровень "1" )
            unsigned    PHY_LB          :1;     // Включение тестового замыкания ПРД на ПРМ на выходе
                                                   контроллера PHY-уровня до аналоговой части ПРМ/ПРД
                                                   (активный уровень "1" )
#endif  //__BIT_ORDER_REVERSE__
} t_bits_PHY_CTRL;

typedef struct  bits_PHY_STAT
{
#ifdef  __BIT_ORDER_REVERSE__
    volatile unsigned   PHY_EXT_CRS     :1;     // Индикатор от внешнего контроллера PHY-уровня о
                                                   наличии несущей в линии
    volatile unsigned   PHY_EXT_COL     :1;     // Индикатор от внешнего контроллера PHY-уровня о
                                                   наличии коллизии в линии
    volatile unsigned   reserved3       :3;     // (зарезервировано)
    volatile unsigned   PHY_EXT_LINK    :1;     // Индикатор от внешнего контроллера PHY-уровня о
                                                   наличии подключения в линии
    volatile unsigned   reserved2       :2;     // (зарезервировано)
    volatile unsigned   PHY_INT_CRS     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                   наличии несущей в линии
    volatile unsigned   PHY_INT_COL     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                   наличии коллизии в линии
    volatile unsigned   PHY_INT_LINK    :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                   наличии подключения в линии
    volatile unsigned   PHY_INT_POL     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                   смене полярности сигналов в линии ПРМ
    volatile unsigned   PHY_INT_JAB     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                   превышении времени передачи максимально разрешённой
    volatile unsigned   PHY_INT_JAM     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                   передаче JAM последовательности в случае коллизии
    volatile unsigned   reserved1       :2;     // (зарезервировано)
```

```
#else   //__BIT_ORDER_REVERSE__
    volatile unsigned   reserved1       :2;     // (зарезервировано)
    volatile unsigned   PHY_INT_JAM     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                  передаче JAM последовательности в случае коллизии
    volatile unsigned   PHY_INT_JAB     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                  превышении времени передачи максимально разрешённой
    volatile unsigned   PHY_INT_POL     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                  смене полярности сигналов в линии ПРМ
    volatile unsigned   PHY_INT_LINK    :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                  наличии подключения в линии
    volatile unsigned   PHY_INT_COL     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                  наличии коллизии в линии
    volatile unsigned   PHY_INT_CRS     :1;     // Индикатор встроенного контроллера PHY-уровня о
                                                  наличии несущей в линии
    volatile unsigned   reserved2       :2;     // (зарезервировано)
    volatile unsigned   PHY_EXT_LINK    :1;     // Индикатор от внешнего контроллера PHY-уровня о
                                                  наличии подключения в линии
    volatile unsigned   reserved3       :3;     // (зарезервировано)
    volatile unsigned   PHY_EXT_COL     :1;     // Индикатор от внешнего контроллера PHY-уровня о
                                                  наличии коллизии в линии
    volatile unsigned   PHY_EXT_CRS     :1;     // Индикатор от внешнего контроллера PHY-уровня о
                                                  наличии несущей в линии
#endif  //__BIT_ORDER_REVERSE__
} t_bits_PHY_STAT;

typedef union   MAC_CTRL
{
            unsigned int        all;
            t_bits_MAC_CTRL     bit;
} t_MAC_CTRL;

typedef union   MAC_COLLCONF
{
            unsigned int        all;
            t_bits_MAC_COLLCONF field;
} t_MAC_COLLCONF;


typedef union   INT_SOURCE
{
    volatile unsigned int       all;
            t_bits_MAC_INT      bit;
} t_INT_SOURCE;

typedef union   INT_MASK
{
            unsigned int        all;
            t_bits_MAC_INT      bit;
} t_INT_MASK;

typedef union   GCTRL
{
            unsigned int        all;
            t_bits_GCTRL        bit;
} t_GCTRL;

typedef union   PHY_CTRL
{
            unsigned int        all;
            t_bits_PHY_CTRL     bit;
} t_PHY_CTRL;

typedef union   PHY_STAT
{
    volatile unsigned int       all;
            t_bits_PHY_STAT     bit;
} t_PHY_STAT;



typedef struct  MAC
{
            t_MAC_CTRL          MAC_CTRL;           // Регистр управления MAC-уровнем контроллера
            t_MAC_PACKETLEN     PACKETLEN;          // Регистр управления границами допустимых длин
                                                      пакетов(MinFrame и MaxFrame)
            t_MAC_COLLCONF      COLLCONF;           // Регистр управления обработки коллизий
            unsigned int        IPGT;               // Регистр задания межпакетного интервала
            unsigned int        MAC_ADDR[3];        // Регистр задания MAC-адреса контроллера
            unsigned int        HASH[4];            // Регистр задания HASH-таблицы для расширенной
                                                      фильтрации MAC-адресов
            t_INT_MASK          INT_MASK;           // Регистр маскирования прерываний
            t_INT_SOURCE        INT_SOURCE;         // Регистр флагов прерываний
            t_PHY_CTRL          PHY_CTRL;           // Регистр управления PHY-уровнем контроллера
            t_PHY_STAT          PHY_STAT;           // Регистр состояния PHY-уровня контроллера

             unsigned int       RXBF_HEAD;          // Регистр "головы" буфера ПРМ
    volatile unsigned int       RXBF_TAIL;          // Регистр "хвоста" буфера ПРМ
             unsigned int       TXBF_HEAD;          // Регистр "головы" буфера ПРД (зарезервировано)
    volatile unsigned int       TXBF_TAIL;          // Регистр "хвоста" буфера ПРД (зарезервировано)

    volatile unsigned int       STAT_RX_ALL;        // счетчик кол-ва входящих пакетов дошедших до
                                                      MAC-уровня
```

```
    volatile unsigned int       STAT_RX_OK;         // счетчик кол-ва успешно принятых входящих
                                                       пакетов (!!! при выставленном бите
                                                       ERROR_FRAME_EN - считает все пакеты)
    volatile unsigned int       STAT_RX_OVF;        // счетчик кол-ва входящих пакетов, вызвавших
                                                       переполнение буфера ПРМ
    volatile unsigned int       STAT_RX_LOST;       // счетчик кол-ва входящих пакетов, потерянных из-
                                                       за неготовности MAC-уровня к приёму
                                                       (неготовность дескриптора)
    volatile unsigned int       STAT_TX_ALL;        // счетчик кол-ва исходящих пакетов
    volatile unsigned int       STAT_TX_OK;         // счетчик кол-ва успешно отосланных исходящих
                                                       пакетов

             unsigned int       base_MAC_RxBF;      // Указатель начала буфера ПРМ в адресном
                                                       пространстве контроллера (default = 0x0000)
             unsigned int       base_MAC_TxBF;      // Указатель начала буфера ПРД в адресном
                                                       пространстве контроллера (default = 0x1000)
             unsigned int       base_MAC_RxBD;      // Указатель начала таблицы дескрипторов ПРМ в
                                                       адресном пространстве контроллера (default =
                                                       0x0800)
             unsigned int       base_MAC_TxBD;      // Указатель начала таблицы дескрипторов ПРМ в
                                                       адресном пространстве контроллера (default =
                                                       0x1800)
             unsigned int       base_MAC_RG;        // Указатель расположения области регистров
                                                       управления контроллера (default = 0x1FC0)
    t_GCTRL                     GCTRL;              // Регистр управления внешними интерфейсами
                                                       контроллера
} t_MAC;

#endif  // __MAC_TYPES__
```

ПРИЛОЖЕНИЕ 6. Фильтрация по HASH таблице

В контроллере 5600ВГ1 имеется HASH таблица для осуществления фильтрации входящих пакетов по групповому MAC-адресу.

HASH таблица выполняет роль маски для HASH-функции от MAC-адреса пакета.

HASH-функция вычисляется как 1сдвинутая на значение равное 6 наиболее значащим битам CRC от MAC-адреса назначения пакета.

А именно:

HASH(MAC) = 1<<N.hash, (1)
N.hash = CRC.inv[31:24], (2)
CRC.inv[i] = ~CRC32(MAC)[8-i], (3)*
где i=0 .. 7 — номер бита в байте
CRC32(MAC) — CRC32 от MAC-адреса назначения пакета по полиному 0x04C11DB7 (инв 0xEDB88320)

Примечание:

* если для вычисления используется алгоритм с инверсным CRC, то данный шаг может быть опущен.

Пример:

Пусть имеется пакет с MAC-адресом назначения:
01_23_45_67_89_AB (0x01, 0x23, 0x45, 0x67, 0x89, 0xAB)

Его CRC32 по полиному будет 0x1A0BE8B6,
инверсия производится побайтно по формуле (3):
инверсное значение CRC.inv = 0xA72FE892,
от полученного значения берутся 6 старших бит:
N.hash = CRC.inv[31:26] = (101001b) = 0x19 = 41.

HASH(MAC) = 1<<N.hash = 1<<41 = 0x0000_0200_0000_0000

Таким образом, при установке 41-го бита HASH-таблицы (значение 0x0000_0200_0000_0000) будет разрешен прием пакета с MAC-адресом 01_23_45_67_89_AB, а так же всех пакетов CRC32, MAC-адрес назначения которых будет иметь старшие 6 разрядов равными 101001b.

Лист регистрации изменений

№ п/п	Дата	Версия	Краткое содержание изменения	№№ изменяе-мых листов
1	25.01.2010	1.3	1. Исправлена схема подключения; 2. Таблица 6 приведена в соответствие с ТУ; 3. Введен лист регистрации изменений	17 28
2	01.02.2010	1.4	Значения параметров U_{IL}, U_{IH}. Описание выводов, вывод 18.	4, 28
3	05.04.2010	1.5	Исправлена таблица 2. Дополнен раздел «Контроль MAC-адреса» Исправление ошибок Таблица 2.1: - исправлены ошибки, - дополнен раздел 2.2 MAC_CTRL, - исправлен раздел 2.3 COLLCONF, - дополнен раздел 2.5 INT_MSK/ INT_SRC. Исправлена таблица 6, таблица 7-приведены в соответствие с ТУ.	6, 7 15 13, 18 21 22 23 28, 29
4	27.04.2010	1.6	Замена логотипа	1
5	18.06.2010	1.7	Добавлены зависимости	
6	09.07.2010	2.0	Корректировка по этапу выполнения ОКР	-
7	30.11.2010	2.1	Добавлены правила подключения неиспользуемых проводов (рис.5)	47
8	27.01.2011	2.2	Приведено в соответствие с ТУ	7, 8, 10, 11, 19
9	02.03.2011	2.3	Приведено в соответствие с ТУ и РЭ	По тексту
10	22.04.2011	2.4	Уточнение раздела «Описание функционирования микросхемы» и Приложения 1	По тексту
11	12.10.2011	2.5	Уточнение наименования микросхем	По тексту
12	01.06.2012	2.6.0	1. Замена таблицы 6; 2. Корректировка таблицы 7; 3. Замена рис.5	45 46 50
13	07.06.2012	2.6.1	Исправление ошибки в таблице 6	45
14	23.11.2012	2.6.2	Приведение в соответствие с ТУ после пересмотра. Редактирование текста	По тексту
15	27.02.2013	2.7.0	Приведено в соответствии с ТУ. Устранены ошибки на стр.12 и 40.	12, 40
16	01.11.2013	2.8.0	Исправлено название разряда регистра флагов	20
17	17.02.2015	2.9.0	Корректировка по замечаниям разработчика	По тексту
18	21.01.2016	2.10.0	Исправления в подразделе «Расширенные настройки передачи пакетов». Исправления в подразделе «Настройка размеров пакетов». Исправлена схема на рисунке 10.	19 19 29
19	22.09.2016	2.11.0	Исправлена схема на рисунке 10. Добавлен раздел «Справочные данные». Исправления на рисунке 2.	29 31 8
20	03.05.2018	2.12.0	Корректировка и внесение дополнений в Приложение 1. Регистры контроллера	40 – 43