Рекомендации к применению (v1.6)

Модулей
“PCIe–1553UDx”
“LPCIe–1553UDx”
“mPCIe–1553UDx”
“XMC–1553UDx”
“CPCIS–1553UDx”
“LAN–MIL1553UDx”
“USB–MIL1553UDx”

Интерфейс ГОСТ Р 52070-2003
(MIL-STD-1553B)

09.06.2023

ООО “НОВОМАР”

.

Оглавление

Введение

В настоящем документе описана организация циклов интерфейса ГОСТ Р 53070-2003 (MIL-STD-1553B), устройство контроллера шины и методы организации обмена с использованием модулей: PCIe-1553UDx. LPCIe-1553UD2. mPCIe-1553UDx, XMC -1553UDx, CPCIS -1553UDx. LAN-MIL1553UDx, USB-MIL1553UDx- далее “xxxx-1553UDx”.

Принятые сокращения.

КШ - контроллер шины
ОУ – оконечное устройство
МШ – монитор шины
АМШ – адресуемый монитор шины
КС – командное слово
КУ – команда управления
ОС – ответное слово
СД – слово данных
ПО – программное обеспечение
DMA - direct memory access – прямой доступ к памяти

1. Инициализация каналов.

Перед началом работы с модулем необходимо сконфигурировать каждый канал в соответствии с заданным режимом работы.

Перед программированием регистра режима «CTRL_REG_PCI» **в первую очередь конфигурируются все вспомогательные регистры специфичные для своих режимов, а так же регистры DMA и прерываний**. Запись в основные регистры, определяющие дискретность работы таймеров разрешение/запрещение адресов/подадресов и других параметров, определяющих основные функции канала в заданном режиме, рекомендуется производить при значении бита 23 (WORK_EN) регистра «CTRL_REG_PCI» в состоянии равном **‘0’**.

Значение полей RT_T1_RCVCK и RT_T1_TRCK регистра «RTBM_CONF_REG_PCI», а так же полей BC_T1_RCVCK и BC_T2_TRCK регистра «BC_CONF_REG_PCI», могут быть установлены в любой момент времени, но **вступают в силу только в момент перезаписи бита WORK_EN регистра «CTRL_REG_PCI» из состояния ‘0’ в состояние ‘1’**.

Старт работы канала модуля для всех режимов за исключением режима **КШ**, начинается сразу после записи в регистр «CTRL_REG_PCI» значений полей DEV_MODE, BUSA_EN, BUSB_EN, INSTRUM_MODE, DEV_ADDR, ADDR_P и бита WORK_EN в **состоянии ‘1’**. Для запуска режима **КШ**, после инициализации регистра «CTRL_REG_PCI», необходимо записать в регистр «BC_CONF_REG_PCI» значение бита BCSTRT в значении **‘1’**.

КШ начнет выполнение инструкций с адреса указанного в поле SINSTR_PTR регистра «START_ADR_INSTR». После старта **КШ** состояние бита BCSTRT значения не имеет.

Каждый канал рассчитан на работу одновременно с двумя шинами **‘А’** и **‘Б’**. При работе с одной шиной, рекомендуется отключить неиспользуемую шину, допустим, BUSB_EN = **‘0’**.

Для работы в режимах **АМШ** и **ОУ** необходимо установить адрес устройства и бит четности адреса устройства на информационной шине. При чётном количестве единиц в двоичном значении адреса устройства, бит должен быть установлен в значение **'1'**, при нечётном в **'0'**, (дополнение до нечётности).

Например: DEV_ADDR = ‘00101’, количество единиц две, соответственно ADDR_P = '1'.

При несоблюдении этого требования, устройство не будет отвечать по информационной линии, но на каждую принятую команду будет выдавать блок DMA с битом RT_PARERR = **'1'** (Ошибка чётности адреса **ОУ**) в слове ошибки.

После старта работы канала модуля (для всех режимов: **КШ**, **ОУ**, **АМШ**, **МШ**), повторная запись любых значений в регистр «CTRL_REG_PCI» с битом WORK_EN в состоянии **‘1’**, не оказывает влияния на режим работы, адрес и выбор шин для данного канала. Останов работы канала осуществляется путем записи в регистр «CTRL_REG_PCI» любого значения с битом WORK_EN в состоянии **‘0’**.

2. Общее понятие о цикле и подциклах интерфейса ГОСТ Р 52070-2003 (MIL-STD-1553B).

Большинство мультиплексных электронных систем работают с постоянным циклом выполнения операций. Основной цикл делится на подциклы (рис.1).
Подциклы в свою очередь состоят из:

Синхронизация подцикла: осуществляется одной из двух команд управления (КУ) «Синхронизация» или сообщением блока данных. Контроллер шины (**КШ**) может информировать подсистему о начале и номере подцикла, однако, в большинстве случаев, синхронизация подцикла не используется.

Периодические сообщения: являются запланированными (критичными по времени), они имеют строго определенное время отправки, запланированное в выделенных для них подциклах.

Непериодические сообщения: используются при возникновении определенных условий в подсистеме и используются для разрешения этих условий.
Например: обработка запроса на обслуживание ОУ.

Коррекция ошибок, запрос статусов, передача групповых сообщений: во время этого периода времени **КШ** обрабатывает ошибки, возникшие во время подцикла. **КШ** также может опрашивать оконечные устройства (**ОУ**), имеющие запрос на обслуживание. Этот интервал времени также может быть использован **КШ** для передачи КУ или групповых КУ. Эта часть временного интервала может не использоваться как часть запланированной передачи данных или иметь низкоприоритетную (по мере возможности выполнения) функцию.

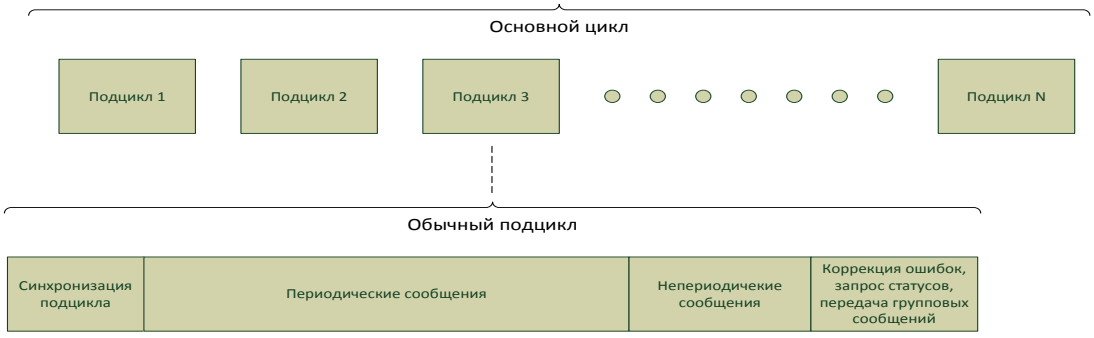

Рисунок 1. Основной цикл интерфейса ГОСТ Р 52070-2003 (MIL-STD-1553B).

3. Распределение памяти КШ и работа модулей в режиме КШ.

Подробное описание работы модуля в режиме **КШ** находится в «Руководство по программированию модулей “PCIe–1553UDx”,“XMC–1553UDx”,“CPCIS–1553UDx”,“mPCIe–1553UDx” интерфейса ГОСТ Р 52070-2003 (MIL-STD-1553B)».

Рассмотрим распределение основных областей памяти **КШ** применительно к нескольким подциклам (рис.2):

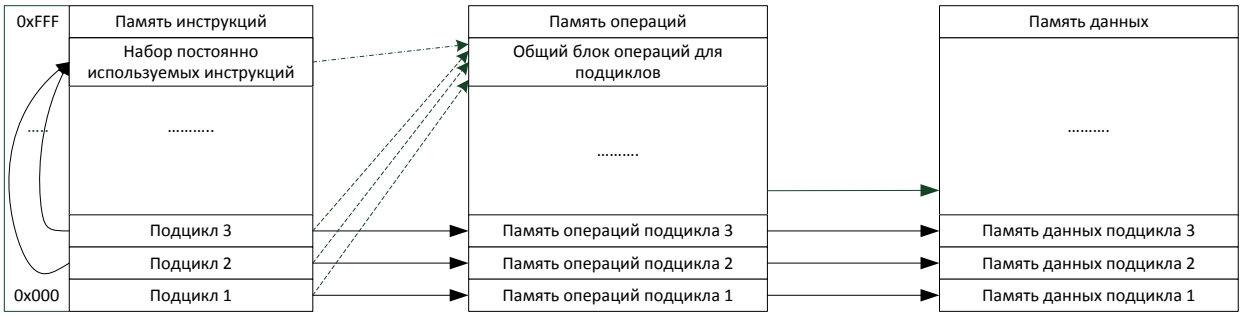

Рисунок 2. Распределение основных областей памяти КШ применительно к нескольким подциклам.

В данной иллюстрации обращается внимание на то, что все области памяти разделены на блоки, соответствующие подциклам. В то же время, память операций может содержать общий блок для всех подциклов, если код операции постоянен (например, для вычитывания данных постоянной длины из одних и тех же **ОУ**, опросе статуса конкретных **ОУ** и т.п.). Также обращается внимание на то, что в памяти инструкций может быть выделена область для постоянно используемого набора команд в подциклах. Распределение памяти инструкций внутри подцикла приведено на рис.3.

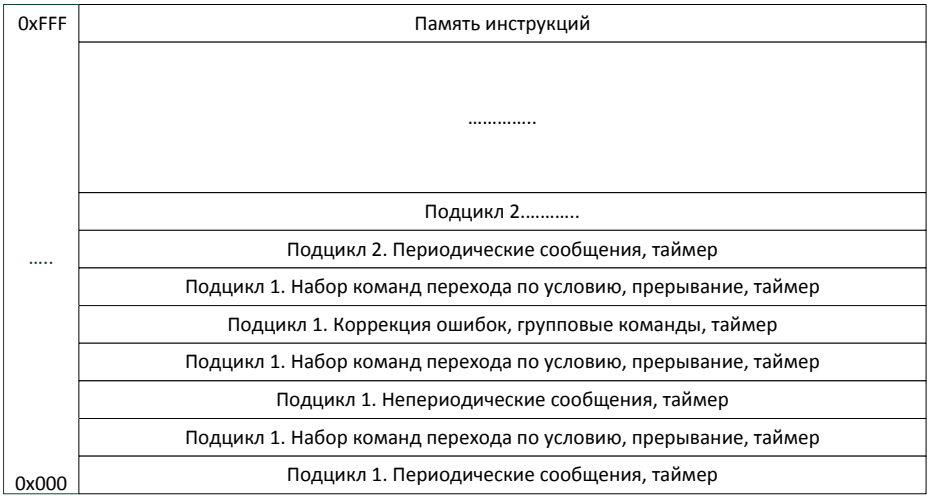

Рисунок 3. Распределение памяти инструкций внутри подцикла.

Такая организация распределения областей памяти совместно с командами перехода по условию позволяет организовать автономную работу системы без дополнительного перепрограммирования. В конце каждого этапа каждого подцикла возможен запуск внутренних таймеров, установка прерывания по условию на шину PCIe, ввод цикла ожидания по условию ожидания флага и т.п. В большинстве случаев перепрограммирование необходимо только в области передаваемых данных.

После каждого этапа, представленного на рис.3 и большом объеме набора инструкций возможно полное перепрограммирование связанных с ним областей памяти, а также освободившейся областью памяти инструкций. Также имеется возможность автоматического повтора ошибочного сообщения (до 2х раз), с возможностью автоматической передачи данного сообщения по резервной шине, что существенно упрощает программное обслуживание.

- Аппаратные задержки, вносимые автоматом **КШ** между записью в регистры запуска или флагов через шину PCIe, и началом передачи данных не превышают **1,5 мкс**.
- Аппаратные задержки между последовательностью инструкций с передачей командного слова (КС) или КУ не превышают **6 мкс**.
- Время выполнения внутренних инструкций (без КС, КУ и ожидания таймеров) автоматом **КШ** не более **150 нс** (в зависимости от типа и условия инструкции).

4. Инструкции КШ.

Все инструкции **КШ** можно разделить на 9 групп:

- **Инструкции для передачи сообщений.** Предназначены для передачи основных форматов сообщений в соответствии с ГОСТ Р 52070-2003 п. 4.5 «Форматы сообщений».
  XEQ, XQG - код условия проверяется перед выполнением
  XQF, XFG - код условия проверяется после выполнения, в случае выполнения условия при следующем выполнении инструкции, расположенной по этому же адресу, будет выполнено сообщение из памяти операций с адресом равным: первоначальное значение параметра XOR 002h (000h → 002h → 000h). Если код условия не удовлетворяется поле параметра остаётся неизменным.
  - инструкции, следующие за инструкциями XEQ и XQF, будут выполнены только после того как значение счетчика T2NXT достигнет значения 0.
  - инструкции, следующие за инструкциями XQG и XFG, будут выполнены сразу, после окончания сообщения, значение T2NXT будет проигнорировано.
- **Внутренние инструкции**, обеспечивающие переходы внутри памяти инструкций. Предназначены для перехода по заданному адресу инструкций, вызова подпрограммы и возврат из подпрограммы к адресу вызова.
  JMP, CAL, RTN
- **Инструкции взаимодействия с ПК.** Предназначены для изменения флагов GPF, обеспечивающих возможность ПК изменять ход выполнения инструкций, а так же устанавливать прерывание на шине PCIe.
  FLG, IRQ
- **Инструкции ввода задержки выполнения следующей команды.** Предназначены для запуска таймеров задержки T2NXT и T2ENDF, обеспечивающих задержку выполнения инструкций до конца работы таймеров.
  DLY, WFT
- **Инструкции сравнения текущих значений таймеров и установки флагов.** Предназначены для сравнения текущих значений таймеров T2ENDF и T2NXT со значениями поля параметров, и установкой флагов LT_GPF0 и EQ_GPF1,
  CFT, CMT
- **Инструкции загрузки основного таймера КШ.**
  LTT, LTH
- **Инструкции кадрового таймера.** Предназначены для загрузки и запуска таймера T2ENDF.
  LFT, SFT
- **Инструкции для организации внутреннего цикла выполнения инструкций.** Предназначены для обеспечения декремента заданной ячейки памяти области данных, установки ее адреса и загрузки значения. Совместно с последующей за DSZ инструкцией JMP позволяют выполнить N повторяющихся циклов.
  DSZ, WMP, WMI
- **Инструкция останова автомата КШ.** Предназначена для обеспечения отладки, и корректной остановки автомата **КШ**.
  HLT

5. Команды КШ.

Команды **КШ** предназначены для управления формированием кода условия, в результате выполнения инструкций передачи сообщений, а так же разрешением повтора передачи сообщений и переключением шин:

- TXTTMC17 - управление КУ17 "Синхронизация с СД"
- MEMASK - маска признака "Ошибка в сообщении" в ОС ОУ, влияет на коды условия MSKSTATSET, GDBT, FMTERR, BADMSG
- SRQMASK - маска признака "Запрос на обслуживание" ОС ОУ, влияет на код условия MSKSTATSET
- BSYMASK - маска признака "Абонент занят" в ОС ОУ, влияет на коды условия MSKSTATSET, GDBT, FMTERR, BADMSG
- SSYSMASK - маска признака "Неисправность абонента" в ОС ОУ, влияет на код условия MSKSTATSET,
- TFMASK - маска признака "Неисправность ОУ" в ОС ОУ влияет на код условия MSKSTATSET
- RSVMASK - маска резервных бит ОС ОУ влияет на код условия MSKSTATSET
- RTRYENA - разрешение повтора сообщения.
- USEBUSB - использовать шину А/Б
- MASKBCR - маска признака "Принята групповая команда", влияет на код условия MSKSTATSET
- FORMAT - определяет формат сообщения согласно ГОСТ 52070-2003

6. Примеры использования инструкций КШ.

Приведенный ниже пример (Таблица 1 – Таблица 4) иллюстрирует организацию цикла/подцикла с использованием кадрового таймера для установки времени подцикла. Набор инструкций обеспечивает вызов подпрограммы подцикла «MINOR1», а также подпрограммы «NXTFRAME», обеспечивающей, в свою очередь, ожидание окончания и перезагрузку кадрового таймера. Для управления временем подцикла также могут быть использованы поля «Время до начала следующей операции» памяти операций или инструкция DLY. Адреса инструкций приведены во внутреннем формате.

Таблица 1. Пример организации цикла

Адрес	Инструкция	Условие	Параметр	Комментарий
0x000	LFT	Always	0x000A	1000 мкс, подпрограмма MINOR1 вызывается 3 раза => 3 мс, основной цикл
0x001	SFT	Always	0x0000	Запуск кадрового таймера
0x002	CAL	Always	0x0009	Вызов подпрограммы MINOR1
0x003	CAL	Always	0x0010	Вызов подпрограммы NXTFRAME
0x004	CAL	Always	0x0009	Вызов подпрограммы MINOR1
0x005	CAL	Always	0x0010	Вызов подпрограммы NXTFRAME
0x006	CAL	Always	0x0009	Вызов подпрограммы MINOR1
0x007	WFT	Always	0x0000	Ожидание окончания кадрового таймера
0x008	JMP	Always	0x0000	Переход к началу цикла

Таблица 2. Пример организации подцикла «MINOR1»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x009	XEQ	Always	MSG1	Сообщение из области операций MSG1
0x00A	XEQ	Always	MSG2	Сообщение из области операций MSG2
0x00B	XEQ	Always	MSG3	Сообщение из области операций MSG3
0x00C	RTN	Always	0x0000	Возврат из подпрограммы

Таблица 3. Подпрограмма обслуживания кадрового таймера «NXTFRAME»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x010	WFT	Always	0x0000	Ожидание окончания кадрового таймера
0x011	LFT	Always	0x000A	Перезагрузка времени подцикла (1000 мкс)
0x012	SFT	Always	0x0000	Запуск кадрового таймера
0x013	RTN	Always	0x0000	Возврат из подпрограммы

Таблица 4. Альтернативная подпрограмма обслуживания кадрового таймера «NXTFRAME»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x010	CFT	Always	0x0001	Сравнение кадрового таймера со значением 100 мкс
0x011	JMP	LT_ GPF0	0x0010	Переход к началу сравнения, если таймер > 100 мкс
0x012	LFT	Always	0x000A	Перезагрузка кадрового таймера
0x013	SFT	Always	0x0000	Запуск нового цикла кадрового таймера
0x014	RTN	Always	0x0000	Возврат из подпрограммы

Возможны схемы конфигурация для отправки как высокоприоритетных, так и низкоприоритетных асинхронных сообщений.

Пример отправки высокоприоритетных сообщений

Пример отправки высокоприоритетных сообщений представлен в таблицах 5 и 6. В данном примере использован модифицированный подцикл «MINOR1».

Перед началом отправки асинхронных сообщений хост (ПО пользователя) должен подготовить и полностью записать необходимые области памяти **КШ** (память операций и память данных), после чего установить в 1 флаг «GPF3».

При выполнении цепочки инструкций и установленном флаге «GPF3», будет произведён переход на подпрограмму «A_HP» и асинхронные сообщения будут отправлены.

По окончании подпрограммы будет сброшен флаг «GPF3».

Таблица 5. Пример применения высокоприоритетного асинхронного сообщения в подцикле «MINOR1»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x009	XEQ	Always	MSG1	Сообщение из области операций MSG1
0x00A	CAL	GPF3	A_HP	Если системой установлен флаг GPF3 (GPF3 = 1), вызов подпрограмы A_HP, для обслуживания высокоприоритетного асинхронного сообщения
0x00B	XEQ	Always	MSG2	Сообщение из области операций MSG2
0x00C	CAL	GPF3	A_HP	Если системой установлен флаг GPF3 (GPF3 = 1), вызов подпрограмы A_HP, для обслуживания высокоприоритетного асинхронного сообщения
0x00D	XEQ	Always	MSG3	Сообщение из области операций MSG3
0x00E	CAL	GPF3	A_HP	Если системой установлен флаг GPF3 (GPF3 = 1), вызов подпрограмы A_HP, для обслуживания высокоприоритетного асинхронного сообщения
0x00F	RTN	Always	0x0000	Возврат из подпрограммы

Таблица 6. Пример подпрограммы обслуживания высокоприоритетных сообщений «A_HP»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x020	XEQ	Always	A_MSG1	Сообщение из области операций A_MSG1
0x021	XEQ	Always	A_MSG2	Сообщение из области операций A_MSG2
0x022	FLG	Always	0x0800	Сброс флага GPF3
0x023	RTN	Always	0x0000	Возврат из подпрограммы

Пример отправки низкоприоритетных сообщений.

Пример отправки низкоприоритетных сообщений представлен в таблицах 7 и 8. В данном примере использована модифицированная подпрограмма «NXTFRAME».

Перед началом отправки асинхронных сообщений хост (ПО пользователя) должен подготовить и полностью записать необходимые области памяти **КШ** (память операций и память данных), после чего установить в 1 флаг «GPF4».

При выполнении цепочки инструкций и установленном флаге «GPF4», а так же времени достаточном для пересылки сообщения, будет произведён переход на подпрограмму «A_LP» и асинхронные сообщения будут отправлены.

По окончании подпрограммы будет сброшен флаг «GPF4».

Таблица 7. Пример подпрограммы обслуживания низкоприоритетных сообщений на основе подпрограммы «NXTFRAME»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x010	CFT	Always	0x0007	Сравнение со значением кадрового таймера
0x011	JMP	LT_GPF0	0x0015	Если времени для выполнения операции недостаточно – переход к ожиданию окончания времени до конца кадра
0x012	CAL	GPF4	A_LP	Сообщение из области операций A_LP при установленном флаге GPF4 = 1
0x013	CFT	Always	0x0007	Сравнение со значением кадрового таймера
0x014	JMP	LT_GPF0	0x0010	Если времени для выполнения операции достаточно – переход к началу подпрограммы
0x015	WFT	Always	0x0000	Ожидание до конца кадрового таймера
0x016	LFT	Always	0x001E	Перезагрузка времени подцикла (3000 мкс)
0x017	SFT	Always	0x0000	Запуск нового цикла кадрового таймера
0x018	RTN	Always	0x0000	Возврат из подпрограммы

Таблица 8. Пример подпрограммы обслуживания низкоприоритетных сообщений «A_LP»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x020	XEQ	Always	AL_MSG1	Сообщение из области операций AL_MSG1
0x021	XEQ	Always	AL_MSG2	Сообщение из области операций AL_MSG2
0x022	FLG	Always	0x1000	Сброс флага GPF4
0x023	RTN	Always	0x0000	Возврат из подпрограммы

Альтернативная реализация асинхронного сообщения с низким приоритетом.

Альтернативная реализация асинхронного сообщения с низким приоритетом описана ниже (Таблицы 9 и 10). Этот метод обеспечивает большую гибкость, но больше полагается на программное обеспечение хоста для поддержания состояния списка сообщений и времени подциклов.

В частности, этот способ основан на том, что программное обеспечение хоста поддерживает список сообщений в каждом вспомогательном подцикле, и, следовательно, время подцикла, оставшееся в конце подцикла, можно рассчитать и сохранить в программном обеспечении.

Для этой реализации код операции CAL с кодом условия, установленным на «Never» (Not Always), сохраняется в конце каждого второстепенного кадра, как показано в таблице 9 ниже.

При необходимости передачи сообщения, код условия операции CAL заменяется на «Always» (в инструкции необходимо инвертировать 2 бита: бит 20 (Not) и бит 31 (P)) и выполняется подпрограмма передачи асинхронных сообщений.

Таблица 9. Пример применения низкоприоритетного асинхронного сообщения в подцикле «MINOR1»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x009	XEQ	Always	MSG1	Сообщение из области операций MSG1
0x00A	XEQ	Always	MSG2	Сообщение из области операций MSG2
0x00B	XEQ	Always	MSG3	Сообщение из области операций MSG3
0x00C	CAL	Never	A_LP	Вызов подпрограммы не осуществляется никогда
При необходимости передачи, условие Never, заменяется на условие Always				
0x00C	CAL	Always	A_LP	Вызов подпрограммы A_LP
0x00D	RTN	Always	0x0000	Возврат из подпрограммы

Реализация подпрограммы передачи низкоприоритетных сообщений с использованием инструкции DSZ.

Реализация подпрограммы передачи низкоприоритетных сообщений с использованием инструкции DSZ представлена в таблице 10. Инструкция DSZ в асинхронном сообщении пытается уменьшить значение (первоначально установленное на 1), сохраненное по первому неиспользуемому адресу в памяти данных.

Если оно уже равно нулю, код инструкции DSZ приведет к пропуску следующего кода операции. Это гарантирует, что сообщения в асинхронном сообщении выполняются только один раз.

В следующий раз, когда хост захочет отправить асинхронное сообщение с низким приоритетом, он должен сначала перезагрузить соответствующую ячейку памяти кодом 0x0001 (адрес ячейки памяти так же можно установить инструкцией WMP, значение – инструкцией WMI).

Таблица 10. Пример подпрограммы обслуживания низкоприоритетных сообщений «A_LP» с использованием инструкции «DSZ»

Адрес	Инструкция	Условие	Параметр	Комментарий
0x020	DSZ	Always	DSZ_ADR1	Адрес в памяти данных DSZ_ADR1
0x021	XEQ	Always	AL_MSG1	Сообщение из области операций AL_MSG1
0x022	DSZ	Always	DSZ_ADR2	Адрес в памяти данных DSZ_ADR2
0x023	XEQ	Always	AL_MSG2	Сообщение из области операций AL_MSG2
0x024	DSZ	Always	DSZ_ADR3	Адрес в памяти данных DSZ_ADR3
0x025	XEQ	Always	AL_MSG3	Сообщение из области операций AL_MSG3
0x026	DSZ	Always	DSZ_ADR4	Адрес в памяти данных DSZ_ADR4
0x027	XEQ	Always	AL_MSG4	Сообщение из области операций AL_MSG4
0x028	DSZ	Always	DSZ_ADR5	Адрес в памяти данных DSZ_ADR5
0x029	XEQ	Always	AL_MSG5	Сообщение из области операций AL_MSG5
0x02A	RTN	Always	0x0000	Возврат из подпрограммы

Реализация второстепенного подцикла по условию.

В некоторых приложениях может возникнуть необходимость выполнить сообщение или второстепенный подцикл по условию.

Для простых приложений с несколькими условными сообщениями в качестве кода условия, для кода операции XEQ или CAL можно легко использовать специальный флаг общего назначения, как показано в предыдущих примерах. Однако, поскольку имеется ограниченное количество доступных флагов общего назначения, может возникнуть необходимость объединить несколько флагов общего назначения, чтобы получить дополнительные коды условия.

В приведенном ниже упрощенном примере в качестве двоичного кодированного десятичного формата используется GPF(4 : 2), позволяющие организовать до 4 условий с использованием только трех флагов общего назначения.

Для этого используется код операции JMP вместе с условием перехода. В этом случае старший значащий бит (GPF4) используется в качестве бита разрешения для ввода набора команд условного ветвления. Если GPF4 не установлен (Not GPF4 = 1), то код вернется к началу условия. Если GPF4 установлен (GPF4 = 1), то контроллер будет оценивать оставшиеся два бита для формирования соответствующего условия в соответствии с таблицей 11.

Следует отметить, что флаги общего назначения GPF0 и GPF1 обычно используются для сравнения таймера кадров и таймера сообщений и могут быть недоступны для пользователя.

Однако реализация расширенной версии схемы, как описано выше, позволит использовать оставшиеся 6 флагов общего назначения, которые могут поддерживать 32 возможных результатов с битом разрешения, или 64 возможных результата, если бит разрешения не используется.

Условные сообщения, показанные в Таблице 11, также могут быть условиями, если используются коды инструкций CAL, а не коды инструкций XEQ.

Таблица 11. Использование условий с применением GPF(4:2)

GPF4	GPF3	GPF2	Условие
1	1	1	4
1	1	0	3
1	0	1	2
1	0	0	1

В таблице 12 показан пример цикла с тремя периодическими сообщениями и до 4 непериодических сообщений, которые можно выполнить, установив определенный битовый шаблон в GPF(4 : 2). Инструкции по адресам 0x02B, 0x02E, 0x031, 0x034 будут выполнены при совпадении условий, определяемых GPF(4 : 2).

Таблица 12. Пример использования GPF(4 : 2).

Адрес	Инструкция	Условие	Параметр	Комментарий
0x020	XEQ	Always	MSG1	Сообщение из области операций MSG1
0x021	XEQ	Always	MSG2	Сообщение из области операций MSG1
0x022	XEQ	Always	MSG3	Сообщение из области операций MSG1
0x023	JMP	GPF4	+2	Переход по адресу 0x025
0x024	JMP	Not GPF4	-4	Переход по адресу 0x020
0x025	JMP	Not GPF3	+2	Переход по адресу 0x027
0x026	JMP	GPF3	+3	Переход по адресу 0x029
0x027	JMP	Not GPF2	+4	Переход по адресу 0x02B
0x028	JMP	GPF2	+6	Переход по адресу 0x02E
0x029	JMP	Not GPF2	+8	Переход по адресу 0x031
0x02A	JMP	GPF2	+10	Переход по адресу 0x034
0x02B	XEQ	Always	C_MSG1	Сообщение из области операций C_MSG1 будет выполнено, если GPF(4:2) = 100
0x02C	FLG	Always	0x1C00	Сброс флагов GPF(4:2)
0x02D	JMP	Always	-13	Переход по адресу 0x020
0x02E	XEQ	Always	C_MSG2	Сообщение из области операций C_MSG2 будет выполнено, если GPF(4:2) = 101
0x02F	FLG		0x1C00	Сброс флагов GPF(4:2)
0x030	JMP	Always	-16	Переход по адресу 0x020
0x031	XEQ	Always	C_MSG3	Сообщение из области операций C_MSG3 будет выполнено, если GPF(4:2) = 110
0x032	FLG		0x1C00	Сброс флагов GPF(4:2)
0x033	JMP	Always	-19	Переход по адресу 0x020
0x034	XEQ	Always	C_MSG4	Сообщение из области операций C_MSG4 будет выполнено, если GPF(4:2) = 111
0x035	FLG		0x1C00	Сброс флагов GPF(4:2)
0x036	JMP	Always	-22	Переход по адресу 0x020

Реализация периодических сообщений.

Во многих приложениях для **КШ** возникает необходимость отправлять одно и тот же сообщение с определенным периодом. В случае данных передаваемых в **ОУ**, механизм **КШ** помещает данные в блоки данных, связанные с каждым сообщением. Программное обеспечение хоста отвечает за заполнение блоков данных.

Для описанного выше случая обычно используется двойная буферизация, чтобы гарантировать согласованность данных. Для контроллера шины, передающего данные, **КШ** может быть сконфигурирован для автоматического «пинг-понга» между двумя блоками данных каждый раз, когда обрабатывается сообщение. Используя инструкцию XQF, **КШ** может быть сконфигурирован для чередования двух блоков передаваемых данных для одного и того же сообщения.

Также возможна передача блока данных длиной больше 32 слов, это достигается размещением в памяти данных двух блоков подряд. В этом случае вначале будет передан первый блок данных, и, непосредственно за ним, следующий блок данных.

В случае выполнения условия после выполнения инструкции, при следующем выполнении этой инструкции, расположенной по тому же адресу, будет выполнено сообщение из памяти операций с адресом равным: первоначальное значение параметра XOR 002h (например: 000h → 002h → 000h).

Если код условия не удовлетворяется поле параметра остаётся неизменным. Чтобы уменьшить общий размер списка команд **КШ** и размер кода памяти инструкций, рекомендуется создать подпрограмму для обработки конкретного сообщения.

Затем подпрограмма может быть выполнена в нескольких второстепенных кадрах или в главном кадре **КШ**. Код инструкции **КШ**, показанный в таблице 13, демонстрирует использование команды XQF для реализации двойной буферизации.

Также данный пример можно использовать для переключения направления приема/передачи сообщений, поскольку сами операции могут быть различны.

Таблица 13. Использование инструкции XQF.

Мнемоника адреса	Инструкция	Условие	Параметр	Комментарий
MSG1XQF	XQF	GPF3	OP_ADDR, OP_ADDR XOR002h	При условии GPF3 =1, после выполнения инструкции поле параметра (адрес памяти операций) станет равным OP_ADDR XOR 002h. При условии GPF3 =0 поле параметра не изменится.
	JMP	BADMSG	MSGERROR	При ошибке в сообщении переход к "MSGERROR"
	RTN	GPF3	0x0000	При условии GPF3 = 1 – автопереключение разрешено, возврат из подпрограммы
	FLG	Always	0x0010	При условии GPF3 = 0 – автопереключение запрещено. Установка флага GPF4, для информации хоста о передачи сообщения.
	RTN	Always	0x0000	Возврат из подпрограммы
MSGERROR	IRQ	Always	0x0000	Установка прерывания, для информирования об ошибке сообщения
	RTN	Always	0x0000	Возврат из подпрограммы

В случае команд **КШ-ОУ** хост полностью контролирует блоки данных, которые будут передаваться в **ОУ**. Как только блок данных записан, хост может переключать указатель блока

данных для инструкции XQF. Для операций **ОУ-КШ**, удобнее использовать инструкцию XEQ, так как принятые данные в любом случае будут переданы в ПК.

Инструкции **КШ** позволяют хосту разрешать повторные попытки для каждого сообщения в отдельности. Регистр конфигурации «BC_CONF_REG_PCI» позволяет пользователю глобально определять поведение повторных попыток, например, сколько повторных попыток необходимо и производить их по альтернативной шине или нет.

Инструкции **КШ** могут использоваться для объединения стратегий повторных сообщений и переключения шин. Если **КШ** определяет, что конкретный **ОУ** вышел из строя, он может автоматически постоянно переключать сообщение с исходной шины на альтернативную шину, сохраняя пропускную способность **КШ**, устраняя необходимость повторной отправки сообщений по неисправным каналам **ОУ**.

Инструкция XQF может использоваться для реализации автономной версии переключения шины.

Пример в таблице 14 показывает, как использовать инструкцию XQF для схемы автономного переключения шин.

Таблица 14. Пример переключения шин с использованием инструкции XQF

Метка адреса	Инструкция	Условия	Параметр	Комментарий
	FLG		0x1000	Инициализация путем сброса флага GPF4 в 0
MSG1	XQF	BADMSG	OP_ADDR, OP_ADDR XOR002h	Передача сообщения по основной шине из адреса OP_ADDR. В случае ошибки, один повтор по альтернативной шине из адреса OP_ADDR XOR 002h
	JMP	GDBT	NEXT	Если сообщение успешно – переход к NEXT
	JMP	GPF4	ERROR	Условный переход на подпрограмму ошибки, если GPF4 установлен в 1. Это индицирует, что сообщение выполнено с ошибкой на обоих шинах.
	FLG	Always	0x0010	Установка флагаGPF4
	JMP	Always	MSG1	Повтор сообщения по альтернативной шине с операцией OP_ADDR XOR 002h
ERROR	IRQ	Always	0x0000	Прерывание ПК о том, что ошибка произошла на обоих шинах
NEXT				Следующее сообщение

Автоматический повтор сообщений (по основной и альтернативной шинам), возможно так же организовать соответствующей установкой бит управления регистра «BC_CONF_REG_PCI».

Биты управления: BCRE, BC2RE, BCR1A, BCR2A.

Реализация счётчиков.

Контроллер шины поддерживает функцию, которая позволяет пользователям реализовывать выполнение цикла **N** - итераций с использованием инструкций DSZ, WMI и WMP. В этом случае фиксированная ячейка памяти используется для хранения значения «N». Адрес ячейки памяти можно изменить с помощью инструкции WMP. Значение ячейки памяти можно изменить с помощью инструкции WMI, или путем прямой записи по этому адресу программой хоста.

Если код условия оценивается как истинный, инструкция WMI запишет указанное значение параметра в адрес памяти, указанный в последней выполненной инструкции WMP. Это позволяет пользователю инициализировать ячейку памяти значением **«N»**.

Для кода операции DSZ, если код условия выполняется, значение, сохраненное по адресу памяти, указанному в слове параметра, уменьшается на единицу. Если новое значение не равно нулю, выполняется следующая инструкция. Если уменьшенное значение равно нулю, следующая инструкция пропускается.

Таблица 15. Пример циклического выполнения программы.

Метка адреса	Инструкция	Условия	Параметр	Комментарий
	WMP	Always	0x3FFF	Установить указатель адреса на ячейку памяти данных по адресу 0x3FFF.
	WMI	Always	0x000A	Установить число итераций (N) равное 10. Цикл будет выполнен 10 раз
	CAL	Always	LOOP	Вызов подпрограммы LOOP
NEXT				Следующее сообщение
LOOP	XEQ	Always	MSG1	Подпрограмма LOOP: Выполнение MSG1
	DSZ	Always	0x3FFF	Уменьшить на единицу значение N и пропустить инструкцию JMP, если N = 0.
	JMP	Always	LOOP	Переход в начало LOOP если N > 0
	RTN	Always	0x0000	Возврат из подпрограммы, если N = 0

Пример простого программного кода для непрерывного приема данных от двух **ОУ** представлен в таблицах 16 и 17:

Таблица 16. Пример инструкций.

Номер бита																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	XEQ					Защитный код					ALWAYS					0000															
0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P	XEQ					Защитный код					ALWAYS					0002															
0	0	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	JMP					Защитный код					Not(GPF2)					0000															
0	0	0	1	1	1	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	HLT					Защитный код					ALWAYS					0000															
P	Инструкция					0	1	0	1	0	Условие					Поле параметра															
Поле инструкции																															
18000: 054F 0000 18004: 054F 0002 18008: 0952 0000 1800C: 1D4F 0000																															

Таблица 17. Пример операций.

Номер бита																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Команда КШ – формат 2																Время до начала следующей операции															
0	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Адр = 07 К = 1 Падр = 11 СД = 32																Поле параметра NA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Команда КШ – формат2																Время до начала следующей операции															
0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Адр = 08 К = 1 Падр = 10 СД = 32																Поле параметра NA															
1C000: 0002 0000 1C004: 3D60 0000 1C008: 0002 0000 1C00C: 4540 0000																															

В данном коде **КШ** будет непрерывно (бесконечно) выдавать в цикле по 2 КС на передачу данных из **ОУ** с адресом 7 и подадресом 11 и **ОУ** с адресом 8 и подадресом 10, в обоих случаях каждый **ОУ** должен передать 32 слова данных.

Остановка программы может быть произведена путем программной установки флага GPF2 в **«1»** функцией **DEVCTL_WR_REG_OFFSET** по адресу 2090h, значение 00040000h (см.Руководство программиста).

Для удобства работы при использовании **КУ** со словами данных или инструкций DSZ, рекомендуется в области памяти данных резервировать постоянную область адресов в начале или конце таблицы данных, достаточную для размещения всех необходимых значений.

7. Системная область данных DMA.

Разбор системной области данных DMA для режима **КШ** и **ОУ** практически одинаков. Произведем разбор на примере **КШ**.

Из первого двойного слова получаем информацию:

- Формат сообщения: определяет количество КС и ОС, а так же их порядок в сообщении;
- Количество служебных слов данных, включая слово данных сопутствующее КУ (команда управления со словом данных) «Значение счетчика»;
- Количество слов данных в сообщении для КС (из командного слова). При ошибочном сообщении данные необходимо игнорировать (нет достоверности).

Количество служебных слов данных принятых по шине рассчитывается по формуле: «Значение счетчика» – (5*2) – 1.

Пример:

*Если поле «Значение счетчика» равно 13, это означает, что в сообщении было 13 – 5*2 - 1 = 2 служебных слова, например КС и ОС (в зависимости от формата сообщения).*

13 – значение счетчика

*5*2 – первые пять служебных двойных слов*

1 – слово ошибки (всегда последнее и всегда присутствует в области служебных данных).

Форматы 1, 2, 4:

КС (КУ);
ОС;
Слово ошибки.

Формат 3:

КС1 (приемник);
КС2 (передатчик);
ОС2 (передатчик);
ОС1 (приемник);
Слово ошибки.

Формат 5:

КУ;
ОС;
СД (слово данных КУ);
Слово ошибки.

Формат 6:

КУ;
СД (слово данных КУ);
ОС;
Слово ошибки.

Форматы 7, 9:
КС (КУ);
Слово ошибки.

Формат 8:
КС1 (приемник, групповая команда);
КС2 (передатчик);
ОС2 (передатчик);
Слово ошибки.

Формат 10:
КУ;
СД (слово данных КУ);
Слово ошибки.

Служебные слова, начиная с двойного слова 6 (слова 11), записываются в порядке их приема по шине. Соответственно, при отсутствии ОС в примере для формата 1, значение счетчика будет равно 12.
Данные будут выглядеть следующим образом:

Пример 1:
Для формата 1;
КС;
Слово ошибки (слово ошибки не будет иметь нулевого значения).

В некоторых случаях, например, при ситуации на шине, определенной в «п.6.5.1 Состояние отсутствия ответа в форматах сообщения **КШ-ОУ** и **ОУ-КШ**» ГОСТ Р 52074-2003, вместо одного сообщения будут приняты два.

Пример 2:

Первое сообщение:
Форматы 1, 2:
КС;
Слово ошибки (ошибка таймаута t1 для режима КШ), для остальных режимов, данная ошибка устанавливается в первом слове DMA.

Второе сообщение:
Формат ?:
КС или КУ (в действительности ОС);
Слово ошибки.

КС или КУ не определено. Поскольку ОС имеет синхросигнал, совпадающий с КС, то и будет воспринят как КС. Интерпретация зависит от того включены ли в ОС один или несколько бит признаков. В слове ошибки для режима КШ будет установлен бит "LFE" (Ошибка самоконтроля данных), означающий, что из линии приняты не ожидаемые (не запрашиваемые) данные, или произошла одновременная (встречная) передача сообщений от нескольких абонентов.
Для остальных режимов в слове ошибки будет установлен бит "MC_ND" (Команда не определена). Соответственно, для всех режимов в первом слове DMA не будет признаков успешного сообщения.

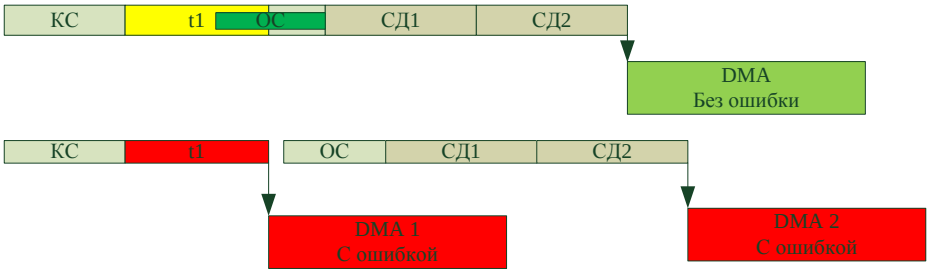

Рисунок 4. Иллюстрация превышения интервала **t1** для примера 2.

Для исключения ситуации встречной передачи данных, рекомендуется устанавливать поле «Время до начала следующей операции» соответствующим суммарному времени, необходимому для передачи всех КС (КУ), ОС, данных, а так же интервалу определяемому значением времени **t1** (RT_T1_RCVCK или BC_T1_RCVCK).

Допустим, для формата 3 (**ОУ-ОУ**) необходимо передать 32 слова данных. Формат сообщения: КС1, КС2, **t1**, ОС2, СД*32, **t1**, ОС1. В данном сообщении насчитывается 36 слов и два интервала **t1**. Длительность каждого слова равна **20мкс**, возьмем стандартный интервал **t1** равным **14мкс**.

В результате длительность всего сообщения равна **748мкс**, соответственно **КШ** начнет следующее сообщение только после окончания этого времени. Без установки поля «Время до начала следующей операции» (значение равно 0), **КШ** выдаст в линию КС1 и КС2, выждет интервал **t1** и, не получив ОС передатчика, сформирует ошибку сообщения, а затем приступит к выполнению следующей операции. Передача сообщения **КШ** начнется только в том случае, если линия будет свободной, в противном случае **КШ** будет ожидать окончание передачи данных с их фиксацией и сообщением об ошибке.

При суммарном значении задержки приема данных > **t1 + t2** (обычно **t1 = 14мкс**, **t2 = 6мкс**), возможна встречная передача данных, в этом случае тип ошибки предугадать невозможно.

8. Пример обработки КУ0 «Принять управление интерфейсом» в режиме КШ.

В случае необходимости использования в сообщениях команды КУ0 «Принять управление интерфейсом» возникает необходимость определения **КШ** подтверждения в ОС **ОУ** приёма команды. Пример инструкций в таблице 18 показывает, как можно автоматически остановить **КШ** без вмешательства ПО.

Таблица 18. Пример обработки КШ КУ0 «Принять управление интерфейсом»

Метка адреса	Инструкция	Условия	Параметр	Комментарий
	……	……	……	
	XEQ	Always	MSG1	Передача КУ0
	JMP	NORESP	ERROR	Если нет ответа – обработка ошибки
	JMP	Not GDBT	ERROR	Если сообщение не успешно – обработка ошибки
	JMP	Not MSKSTATSET	ERROR	Если не установлен бит статуса – обработка ошибки
	IRQ	GDBT	0x0000	Установить прерывание, если данные переданы верно
	HLT	MSKSTATSET	0x0000	Останов выполнения инструкций (остановка КШ)
	JMP	Always	ERROR1	Дополнительная проверка правильности обработки
ERROR	IRQ	Always	0x0000	Прерывание ПК о том, что произошла ошибка
Инструкции обработки ошибки сообщения				
ERROR1	Инструкции обработки ошибки останова			

В данном примере реализована передача сообщения КУ0, проверка получения ОС **ОУ**, проверка того, что в ОС **ОУ** не установлен бит «Ошибка в сообщении», а так же установлен не маскируемый бит «Принято управление интерфейсом». После всех проверок устанавливается прерывание ПК и остановка **КШ**. Так же производится дополнительная проверка правильности остановки **КШ**. Перед выполнением данных инструкций рекомендуется использовать КУ "Передать слово ВСК ОУ", с последующим анализом информации встроенной системы контроля ОУ. Полная остановка **КШ** осуществляется записью бита WORK_EN регистра «CTRL_REG_PCI» в состоянии равном **‘0’.**

9. Пример обработки КУ0 «Принять управление интерфейсом» в режимах ОУ и АМШ.

Для обработки команды КУ0 «Принять управление интерфейсом» в **ОУ** или **АМШ** необходимо предварительно разрешить выполнение этой команды в регистре «MODET_REG» (установить бит 0 в состояние **‘1’**). В этом случае, при получении КУ0, устройство автоматически ответит ОС с установленным в нем признаком «Принято управление интерфейсом», в противном случае в ОС будет установлен признак «Ошибка в сообщении».

При назначении адреса **ОУ** или **АМШ** возможно использование зарезервированного адреса ОУ «0».

Для ускорения реакции системы рекомендуется разрешить прерывание от **ОУ** при получении КУ, в регистрах «RT_INT_REG» и «INTERRUPT MASK». Так же рекомендуется предварительная запись памяти инструкций, операций и данных («BC_INSTR_RAM», «BC_OPERATION_RAM» и «BC_DATA_RAM»), а так же основных регистров КШ.

Не смотря на то, что устройство не находится в режиме КШ, область памяти и регистров КШ не затрагивается при работе в режимах ОУ и АМШ.

Таким образом, после получения КУ0, ПО должно произвести выключение устройства (бит WORK_EN = **‘0’**) и не менее, чем через 1мкс его запуск (бит WORK_EN = **‘1’**), но уже в режиме **КШ**, при предварительно установленных в заданные значения регистрах и памяти **КШ.**

Более подробно порядок инициализации и остановки устройства приведен в п. 1. Инициализация каналов модулей "PCIe-1553UDx", "XMC-1553UDx", "CPCIS-1553UDx", "mPCIe-1553UDx" данного руководства.

10. Организация и работа таймеров.

Внутренняя организация таймеров для каждого канала в режиме **КШ** представлена на рис.5.

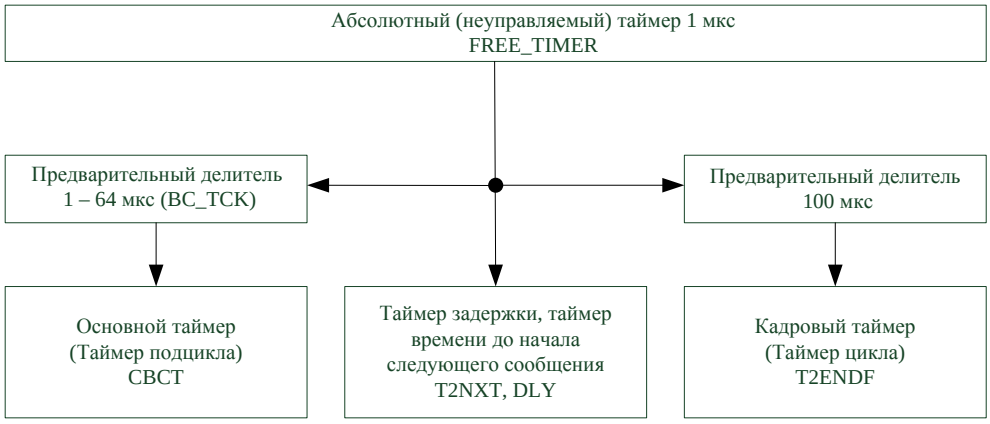

Рисунок 5. Организация таймеров для каждого канала в режиме КШ.

Структура таймеров состоит из:

- Абсолютного (неуправляемого) таймера FREE_TIMER с дискретностью выдачи синхросигнала 1 мкс;
- Кадрового таймера или таймера цикла T2ENDF с предварительным делителем на 100 мкс;
- Основного таймера или таймера подцикла CBCT с предварительным делителем от 1 до 64 мкс;
- Таймера, выполняющего роль задержек T2NXT и DLY и дискретностью 1 мкс.

Рассмотрим связь таймеров на примере организации нескольких циклов (рис. 6)

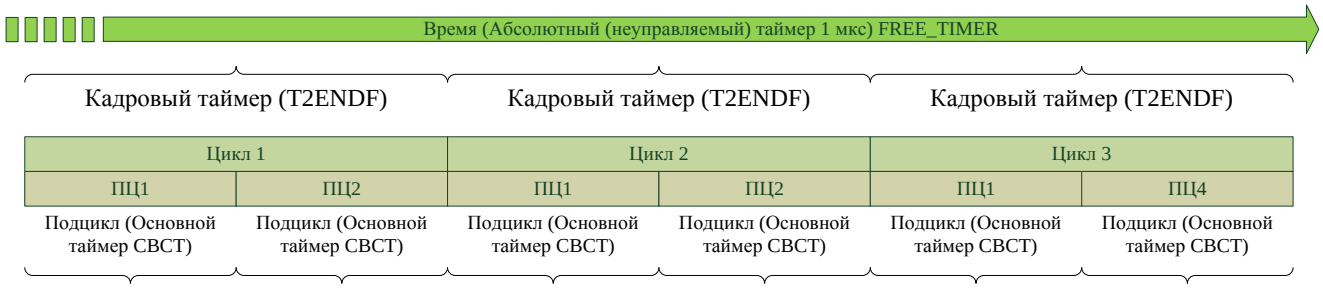

Рисунок 6. Связь таймеров для нескольких циклов.

На данном рисунке приведена связь неуправляемого таймера относительно циклов и подциклов (соответственно кадрового и основного таймеров). Количество подциклов внутри циклов, обозначено условно, и может отличаться для каждого цикла.

Каждое сообщение DMA сопровождается записью значения неуправляемого таймера, что является своеобразной временной меткой и позволяет получить точное значение отправки или приема сообщений вне зависимости от времени цикла или подцикла в любом режиме работы канала. Полученные значения неуправляемого таймера в сообщениях DMA также позволяют синхронизировать значения всех таймеров между собой при работе в реальном масштабе времени.

Программное управление неуправляемым таймером невозможно, сброс таймера осуществляется только по сбросу шины PCIe.

Декремируемый кадровый таймер предназначен для возможности грубого (с дискретностью 100 мкс) отслеживания времени внутри цикла и, соответственно, подциклов.

Загрузка кадрового таймера может быть осуществлена одним из двух способов:

- Записью соответствующего значения в регистр «BC_ FRAME_TIME_REM/ BC_TIME2NEXT» значения T2ENDF, что удобно при отладке программы;
- Использованием инструкции LFT контроллера **КШ**.

Загрузка максимального значения таймера может быть произведена однократно любым из описанных способов. Записанное значение сохраняется во внутреннем регистре таймера.
Запуск таймера осуществляется только инструкцией SFT контроллера **КШ**.
Если загрузка таймера уже произведена, то дополнительная запись в регистр или инструкция LFT, предшествующая инструкции SFT не нужны. При выполнении инструкции SFT таймер перезагрузится записанным значением из собственного внутреннего регистра.
Инструкция WFT позволяет остановить выполнение последовательности инструкций контроллером **КШ** до окончания (обнуления) таймера, после чего продолжается выполнение инструкций, следующих за инструкцией WFT.
Инструкция CFT позволяет сравнить значение таймера с заданной величиной (поле параметра инструкции) и, в результате, установить в соответствующее значение флаги LT_ GPF0 и EQ_GPF1. Значение флагов может быть получено из регистра «GPF/BC_COND_CODE», а также может быть использовано при выполнении условных инструкций контроллером **КШ**.
Время до окончания кадра при запущенном таймере может быть получено из регистра «BC_FRAME_TIME_REM/ BC_TIME2NEXT» (T2ENDF). Данные загруженные в регистр перед запуском таймера сохраняются во внутреннем регистре таймера и для чтения не доступны.

Таймером подцикла является основной таймер **КШ** (СВСТ). Основной таймер предназначен для возможности синхронизации таймеров всей сети (**КШ** и **ОУ**) путем выдачи/приема КУ "Синхронизация" (без СД) или КУ "Синхронизация" (с СД) ГОСТ Р 52070-2003, а также организации тайминга подцикла **КШ**.

Структура основного таймера **КШ** (СВСТ) представлена на рис.7

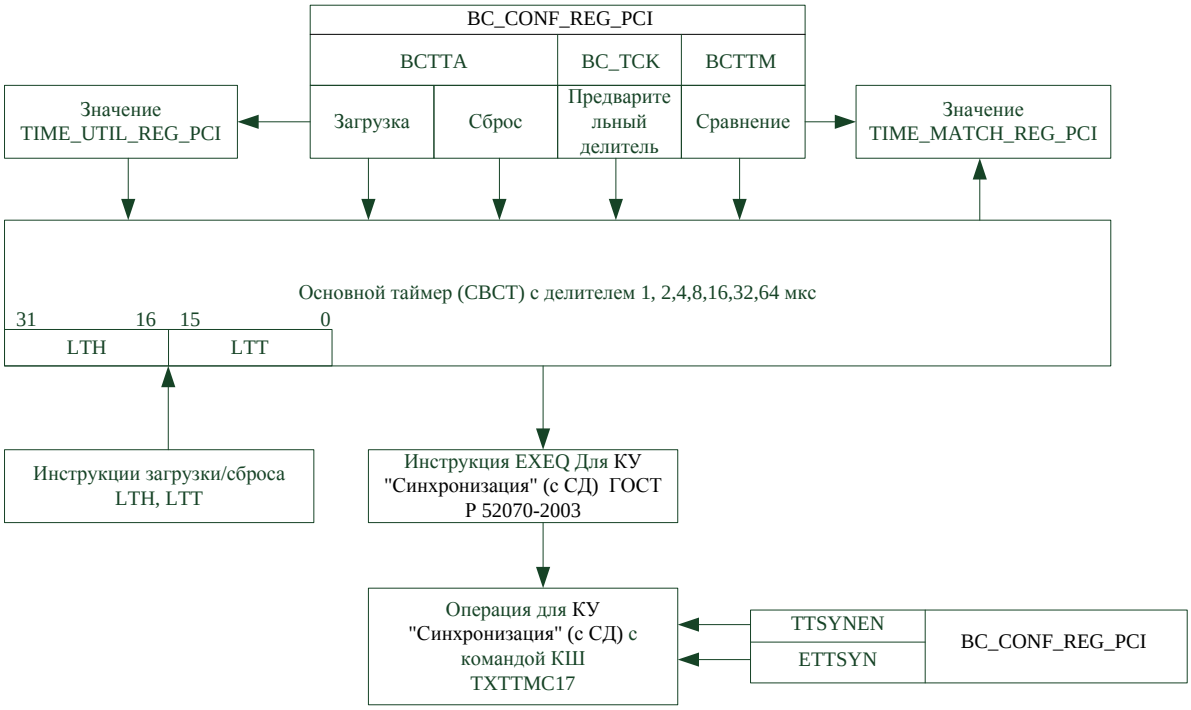

Рисунок 7. Структура основного таймера **КШ** (СВСТ).

Основной таймер **КШ** (СВСТ) является инкремируемым 32-х битным таймером, состоящим из старшей (LTH) и младшей (LTT) половин. Основная конфигурация таймера осуществляется через регистр «BC_CONF_REG_PCI».

Назначение и действие бит регистра:

- Биты BC_TCK устанавливают значение делителя частоты основного таймера и должны быть установлены в заданное пользователем значение до разрешения работы устройства (при значении бита 23 (WORK_EN) регистра «CTRL_REG_PCI» в состоянии равном **‘0’**). После разрешения работы (при значении бита 23 (WORK_EN) регистра «CTRL_REG_PCI» в состоянии равном **‘1’**) перезапись битов BC_TCK не оказывает влияния на делитель частоты;

- Биты BCTTA предназначены для программной загрузки таймера СВСТ значением из регистра «TIME_UTIL_REG_PCI» (загружаются все 32 бита таймера, **BCTTA = “01”**) или сбросом таймера (**BCTTA = “10”**). Остальные значения BCTTA не оказывают влияния на таймер;

- Бит BCTTM = **‘1’** позволяет сравнивать текущее значение таймера со значением, записанным в регистре «TIME_MATCH_REG_PCI».
  В случае равенства или превышения значения регистра однократно (в одном цикле DMA) будет установлен в состояние **‘1’** бит 21 (MTEQ) первого слова DMA системной области данных, так же будет установлен бит 5 (MTEQ) регистра «GPF/BC_COND_CODE».
  При значении бита BCTTM = **‘0’** указанные биты устанавливаться не будут;

- Бит TTSYNEN предназначен исключительно для управлением источником СД команды КУ "Синхронизация" (с СД) ГОСТ Р 52070-2003.
  При установке этого бита в состояние логической **‘1’** источник данных для синхронизации определяет бит 15 команды управления **КШ** (TXTTMC17). Значение TXTTMC17 = **‘0’** - источником слова данных КУ является память данных **КШ**. Значение TXTTMC17 = **‘1’** - источником слова данных КУ являются младшие 16 бит основного

таймера **КШ**. При установке бита TTSYNEN в состояние логического **‘0’** источником слова данных КУ является память данных **КШ** не зависимо от значения бита 15 (TXTTMC17) команды управления **КШ**;

- В случае одновременной установки TTSYNEN и TXTTMC17 в значение **‘1’**, при передаче КУ "Синхронизация" (с СД) ГОСТ Р 52070-2003, **КШ** передаёт значение младших 16 бит основного таймера. При этом если бит ETTSYN регистра «BC_CONF_REG_PCI» установлен в значение **‘1’**, передаваемое СД всегда имеет чётное значение (бит 0 = 0). Значение логического **‘0’** бита ETTSYN не оказывает воздействия на слово данных, т.е. младший бит слова данных соответствует младшему биту значения основного таймера **КШ** и может быть как **‘0’** так и **‘1’**.

В случае переполнения таймера (все 32 бита = 1) однократно (в одном цикле DMA) будет установлен в состояние **‘1’** бит 23 (MT_OVF) первого слова DMA системной области данных, так же будет установлен бит 7 (MT_OVF) регистра «GPF/BC_COND_CODE». При использовании таймера в 32 битном режиме и управлением битами BCTTA регистра «BC_CONF_REG_PCI» рекомендуется пользоваться функцией сброса таймера (BCTTA = **“10”**). При использовании таймера в 16 битном режиме и управлением битами BCTTA регистра «BC_CONF_REG_PCI» рекомендуется пользоваться функцией загрузки таймера (BCTTA = **“01”**), при этом старшие 16 бит регистра «TIME_UTIL_REG_PCI» рекомендуется устанавливать в **‘1’**.

Загрузка и сброс таймера так же управляется инструкциями загрузки/сброса **КШ** LTH и LTT. При использовании таймера в 16 битном режиме рекомендуется использовать инструкцию LTH с полем параметра, содержащем все единицы, при этом инструкции LTH не должна предшествовать инструкция LTT (используется только LTH). Результатом исполнения инструкции LTH будет сброс младших 16 бит таймера в состояние **‘0’**. Для загрузки или сброса таймера в 32 битном режиме инструкции LTT и LTH должны выполняться последовательно (первой LTT второй LTH), в этом случае в таймер будут записаны все 32 бита из полей параметров инструкций. Сброс таймера эквивалентен записи всех нулей из полей параметров обеих инструкций.

Управление загрузкой и сбросом основного таймера **КШ** (СВСТ) может осуществляться как с использованием инструкций LTT и LTH, так и использованием бит управления BCTTA регистра «BC_CONF_REG_PCI». Приоритетным является последнее обращение к основному таймеру. Допустим, таймер был сброшен инструкцией LTH в шестнадцати битном режиме и начал отсчет. До окончания отсчета (переполнения) был произведен сброс таймера битами управления BCTTA регистра «BC_CONF_REG_PCI». Результатом будет обнуление таймера и его перезапуск уже в 32-х битном режиме.

Рассмотрим более подробно условную структуру подцикла (рис.8, рис.9)

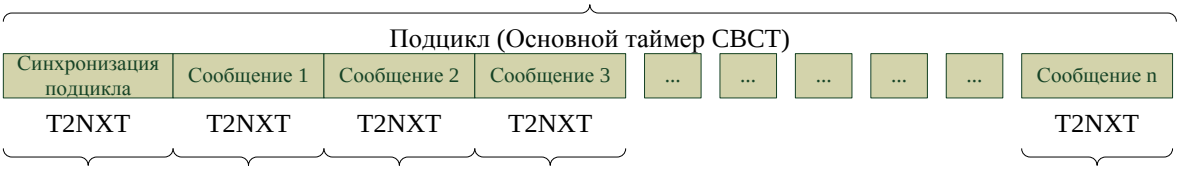

Рисунок 8. Связь таймеров внутри подцикла, состоящего из последовательности сообщений до конца подцикла без интервалов свободного времени.

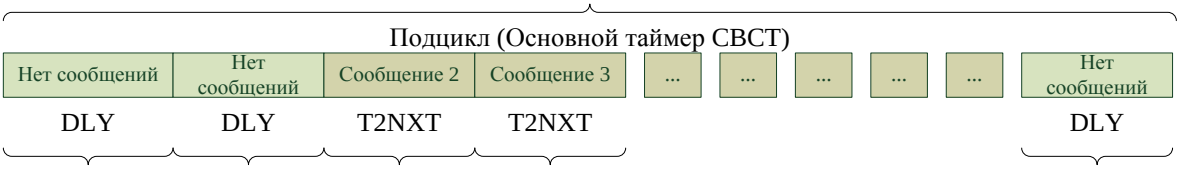

Рисунок 9. Связь таймеров внутри подцикла с заменой неиспользуемых сообщений таймером DLY.

Таймером времени до начала следующей операции (длины сообщения) T2NXT, а так же таймером инструкции DLY является 16-битный декремируемый загружаемый счетчик. Разница между режимами T2NXT и DLY только в источнике загрузки. Для режима T2NXT источником является поле «время до начала следующей операции» выполняемой операции инструкций XEQ и XQF уникальное для каждой операции, для режима DLY - поле параметра самой инструкции.

Существует **три варианта** использования режима T2NXT:

- Минимальное рекомендуемое время рассчитывается в соответствии с форматом сообщения как сумма времени необходимого для передачи всех КС или КУ, СД, ОС, плюс сумма всех таймаутов приёма **t1**, плюс сумма таймаута передачи **t2.** Этот вариант позволяет организовать постоянную длительность подцикла независимо от ошибок в сообщении (отсутствии ответа);

- Минимальное рекомендуемое время рассчитывается как время, необходимое для передачи КС передачи (**ОУ-КШ** или **ОУ-ОУ**), плюс таймаут приёма **t1** для режима **КШ**, плюс дополнительное время для исключения вероятности встречного включения при задержках в линии передачи соизмеримых с таймаутом приёма **t1**.

  Время для передачи КС приема (**КШ-ОУ**), плюс количество передаваемых **КШ** СД, плюс таймаут приёма **t1** для режима **КШ**, плюс дополнительное время для исключения вероятности встречного включения при задержках в линии передачи соизмеримых с таймаутом приёма **t1**.

  Время для передачи КУ без СД или КУ с СД, плюс таймаут приёма **t1** для режима **КШ**, плюс дополнительное время для исключения вероятности встречного включения при задержках в линии передачи соизмеримых с таймаутом приёма **t1**.

  Этот вариант исключает вероятность встречного включения при задержках в линии передачи соизмеримых с таймаутом приёма **t1** и позволяет уменьшить время ожидания в случае КС передачи при отсутствии ответа, но не обеспечивает постоянство длительности подцикла.

- Поле «время до начала следующей операции» выполняемой операции оставляется равным **нулю**.

  Этот вариант не исключает вероятность встречного включения при задержках в линии передачи соизмеримых с таймаутом приёма **t1** и позволяет уменьшить время ожидания в случае КС передачи при отсутствии ответа, но не обеспечивает постоянство длительности подцикла. В случае отсутствия ответа следующее сообщение будет передано **КШ** через время **t1 + t2**.

Описание вариантов использования режима T2NXT будет приведено ниже.

Режим DLY (инструкция DLY) используется в случаях когда необходимо сохранить постоянство длительности подцикла, но по каким-то условиям в данном подцикле, одно или несколько сообщений не передаются. В этом случае значение параметра инструкции DLY рассчитывается также как в первом варианте подсчета режима T2NXT. Таким образом, появляется возможность заменить инструкцию сообщения (XEQ) инструкцией DLY, сохраняя при этом длительность подцикла.

Внутренняя организация таймеров для каждого канала в режимах **ОУ** и **АМШ** представлена на рис.10.

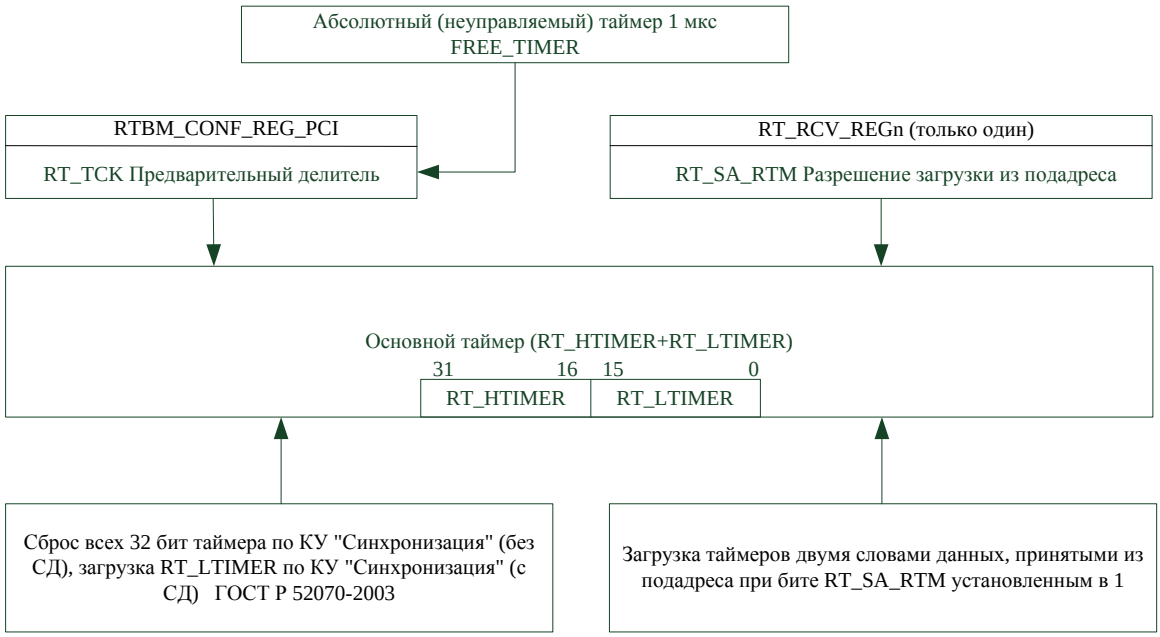

Рисунок 10. Организация таймеров для каждого канала в режимах ОУ и АМШ.

Так же как и в режиме **КШ** синхронизация основного таймера осуществляется абсолютным (неуправляемым) таймером импульсами с периодом 1 мкс. Основной таймер в режимах **ОУ** и **АМШ** управляется непосредственно командами **КШ**, принятыми из линии связи. Это команды "Синхронизация" (без СД) и "Синхронизация" (с СД) ГОСТ Р 52070-2003. Разница в работе таймера при получении этих команд в том, что при получении команды "Синхронизация" (без СД) происходит сброс всех 32 бит таймера, а при получении команды "Синхронизация" (с СД) происходит загрузка младших 16 бит таймера (RT_LTIMER) значением, принятым из СД команды, старшие 16 бит таймера (RT_HTIMER), остаются без изменения. Биты RT_TCK регистра «RTBM_CONF_REG_PCI» устанавливают делитель тактовой частоты таймера в пределах от 1 мкс до 64 мкс. В дополнении к командам **КШ** в **ОУ** есть возможность загрузки всех 32 бит основного таймера двумя СД, принятыми в заданный подадрес. **Для обеспечения этой функции только один из 30 подадресов может иметь включенным бит RT_SA_RTM регистра «RT_RCV_REGn»**.
Данный подадрес может быть выбран персонально для каждого **ОУ**, или быть единым для всех **ОУ** при использовании команд групповой передачи данных.

11. Связь таймера T2NXT с циклами DMA, а так же основным и абсолютным таймерами.

Для понимания длины сообщения, времени начала и конца сообщения рассмотрим более подробно ГОСТ Р 52070-2003: п. 4.5.3.1 «Пауза между сообщениями» и п. 4.5.3.2 «Пауза перед передачей ОС» (рис.11).

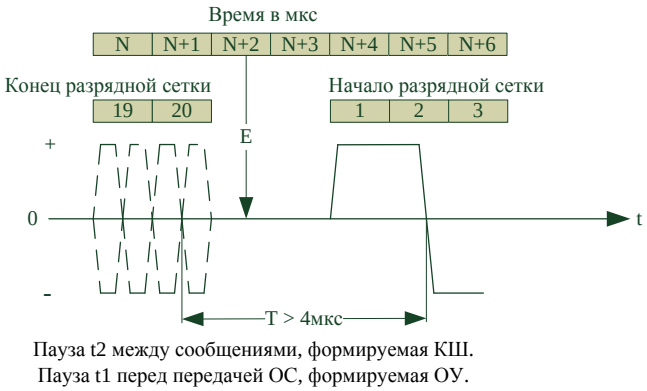

Рисунок 11. Пауза между сообщениями и пауза перед передачей ОС.

Пауза **‘T’** (t1, t2) измеряется полностью от половины последнего бита (бита четности) предыдущего сообщения до половины бита синхросигнала следующего сообщения. Из рисунка видно, что линия передачи находится в состоянии **‘0’** не менее **2-х** микросекунд. Это является защитным интервалом между сообщениями. Конец сообщения определяется по нахождению линии в неактивном положении в течение интервала не менее **0,5** микросекунды после окончания приема последнего бита сообщения (на рисунке обозначено в виде стрелки **«E»**). Соответственно процедура DMA обеспечит передачу данных через шину PCIe не позднее, чем через **1 мкс** после получения последнего бита сообщения. Это так же следует учитывать при подсчете длины сообщения **КШ** (T2NXT). В расчет длины подсчитываемого сообщения следует подставлять не время **t1**, установленное в регистре «BC_CONF_REG_PCI», а время = t1 – 0,5 – 1,5.

Например: в регистре установлено время t1 = 14мкс, в расчет длины сообщения следует подставлять значение t1 = 14 – 0,5 – 1,5 = 12мкс.

Таймаут передачи **КШ t2** уже учитывает эти значения и дополнительных подсчетов не требует. Время T2NXT определяется, как время от половины бита синхросигнала начала текущего сообщения до половины бита синхросигнала начала следующего сообщения.

Существует 5 форматов сообщений с переменной длиной данных рис.12.

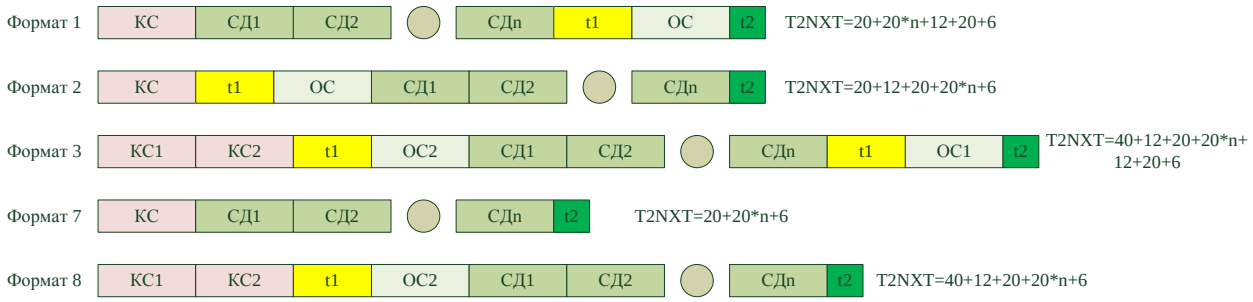

Рисунок 12. Форматы с переменной длиной данных.

Расчет времени T2NXT для каждого формата при **t1 = 14мкс** и **t2 = 6мкс**, установленных для **КШ**, **n** – количество СД (обратите внимание: число СД = 00000 в поле КС соответствует 32 СД):

- Формат 1. T2NXT = 20+20*n+12+20+6 = 58 + 20*n (мкс);
- Формат 2. T2NXT = 20+12+20+20*n+6 = 58 + 20*n (мкс);
- Формат 3. T2NXT = 40+12+20+20*n+12+20+6 = 110 + 20*n (мкс);
- Формат 7. T2NXT = 20+20*n+6 = 26 + 20*n (мкс);
- Формат 8. T2NXT = 40+12+20+20*n+6 = 78 + 20*n (мкс);

Так же существуют 5 форматов с постоянной длиной данных рис.13.

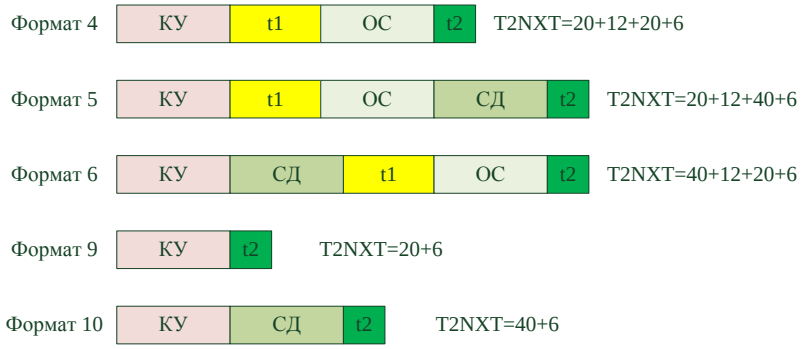

Рисунок 13. Форматы с постоянной длиной данных.

Расчет времени T2NXT для каждого формата при **t1 = 14мкс и t2 = 6мкс**, (параметр по умолчанию) установленных для **КШ**:

- Формат 4. T2NXT = 20+12+20+6 = 58 (мкс);
- Формат 5. T2NXT = 20+12+40 +6 = 78 (мкс);
- Формат 6. T2NXT = 40+12+20+6 = 78 (мкс);
- Формат 9. T2NXT = 20+6 = 26 (мкс);
- Формат 10. T2NXT = 40+6 = 46 (мкс);

Рассмотрим использование таймера T2NXT в различных ситуациях (рис.14, рис.16, рис.17).

На всех трёх рисунках диаграмма **«А»** соответствует максимальному расчетному времени (длине) сообщения с параметрами **t1** и **t2** по умолчанию.
Диаграмма **«Б»** соответствует реальной длине передаваемого сообщения в случае его успешной передачи.
Диаграмма **«В»** соответствует реальной длине передаваемого сообщения в случае его неуспешной передачи.

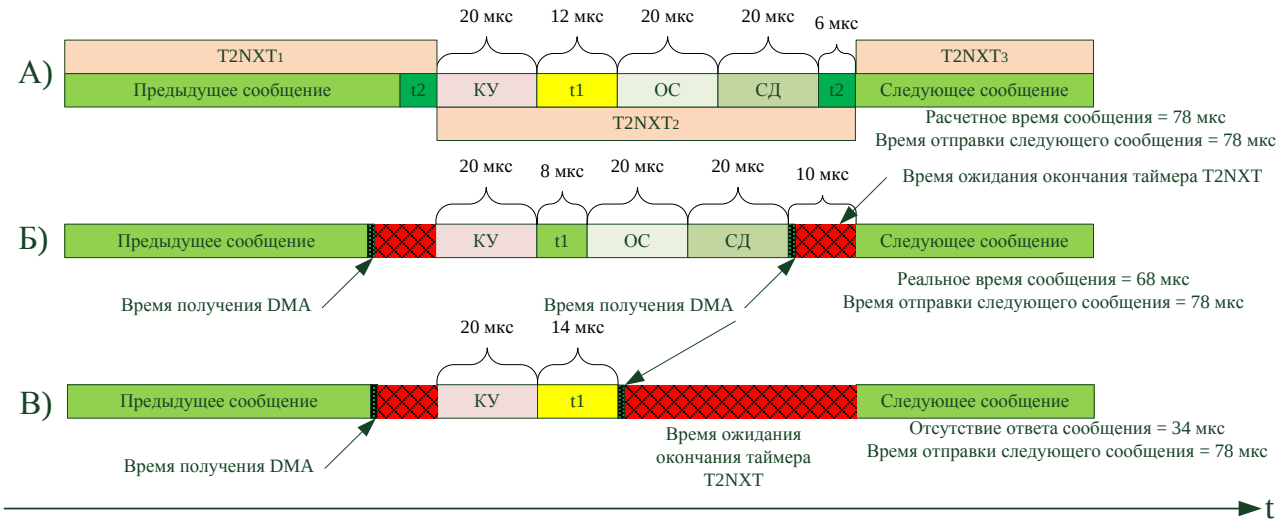

Рисунок 14. Использование таймера T2NXT с параметрами по умолчанию.

На рисунке 14 представлен «форматированный» метод отправки сообщений, когда каждое сообщение отправляется в строго заданное время независимо от результата передачи предыдущего сообщения. Так же из рисунка видно, что время получения DMA и, соответственно, временной метки значения FREE_TIMER (неуправляемый таймер) не совпадают с началом или концом сообщения, что усложняет хронометраж сообщений. Для упрощения привязки сообщений ко времени рекомендуется устанавливать в регистре «BC_CONF_REG_PCI» бит TIME_SEL в значение **‘1’**, что обеспечит выдачу в 5 слове DMA **КШ** время начала сообщения относительно основного таймера (CBCT).

Таким образом:

- Во втором слове DMA **КШ** находится значение FREE_TIMER (неуправляемый таймер), соответствующее концу сообщения;
- В третьем слове DMA **КШ** находится текущее значение декремируемого таймера T2NXT, соответствующее концу передачи сообщения (T2NXTc);
- В пятом слове DMA **КШ** находится время передачи начала сообщения.

В результате, при известной длине сообщения, появляется возможность “связать” основной и абсолютный таймеры, получив достаточно точный хронометраж сообщений. Другими словами, получить время начала и конца сообщения с привязкой к реальному времени, а не только ко времени сети (рис. 15).

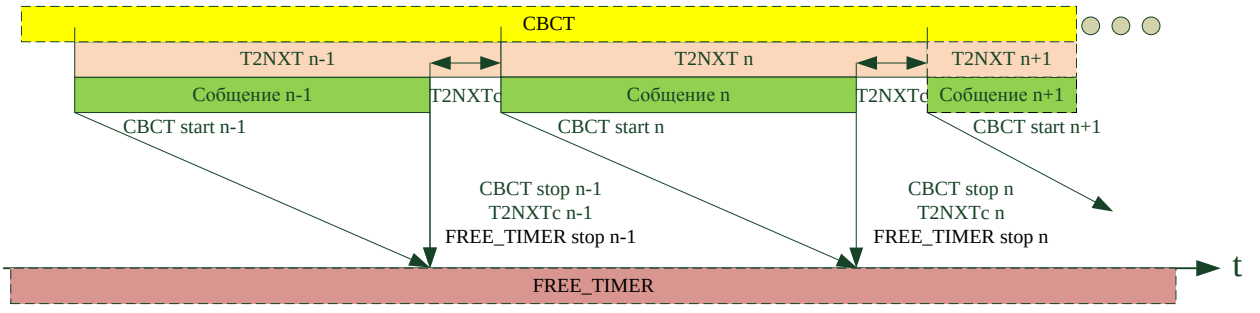

Рисунок 15. Связь таймеров в сообщениях DMA.

Для простой синхронизации времени сети с реальным временем рекомендуется устанавливать в регистре «BC_CONF_REG_PCI» бит TIME_SEL в значение **‘0’**, что обеспечит выдачу в 5 слове DMA **КШ** время конца сообщения относительно основного таймера (CBCT). Поскольку во втором слове DMA **КШ** находится значение FREE_TIMER (неуправляемый таймер), достаточно просто сопоставить эти значения, учитывая полученную разность.

Таким образом:

Для FREE_TIMER:
FREE_TIMER stop n = реальное время конца текущего сообщения (Сообщение n).
FREE_TIMER stop n + T2NXTc n = расчетное время начала следующего сообщения (Сообщение n + 1).

Для старта сообщения CBCT:
CBCT start n = реальное время начала текущего сообщения (Сообщение n).
CBCT start n - T2NXTc n-1 = реальное время конца предыдущего сообщения (Сообщение n - 1).

Для конца сообщения CBCT:
CBCT stop n = реальное время конца текущего сообщения (Сообщение n).
CBCT stop n + T2NXTc n = расчетное время начала следующего сообщения (Сообщение n + 1).

Отсюда:
Расчетное время начала сообщения:
FREE_TIMER stop n + T2NXTc n = CBCT stop n + T2NXTc n + ΔT = CBCT start n+1 + ΔT
Реальное время конца сообщения:
FREE_TIMER stop n = CBCT start n+1 - T2NXTc n + ΔT = CBCT stop n + ΔT
Где ΔT – абсолютная разность начала или конца сообщения между FREE_TIMER и CBCT.

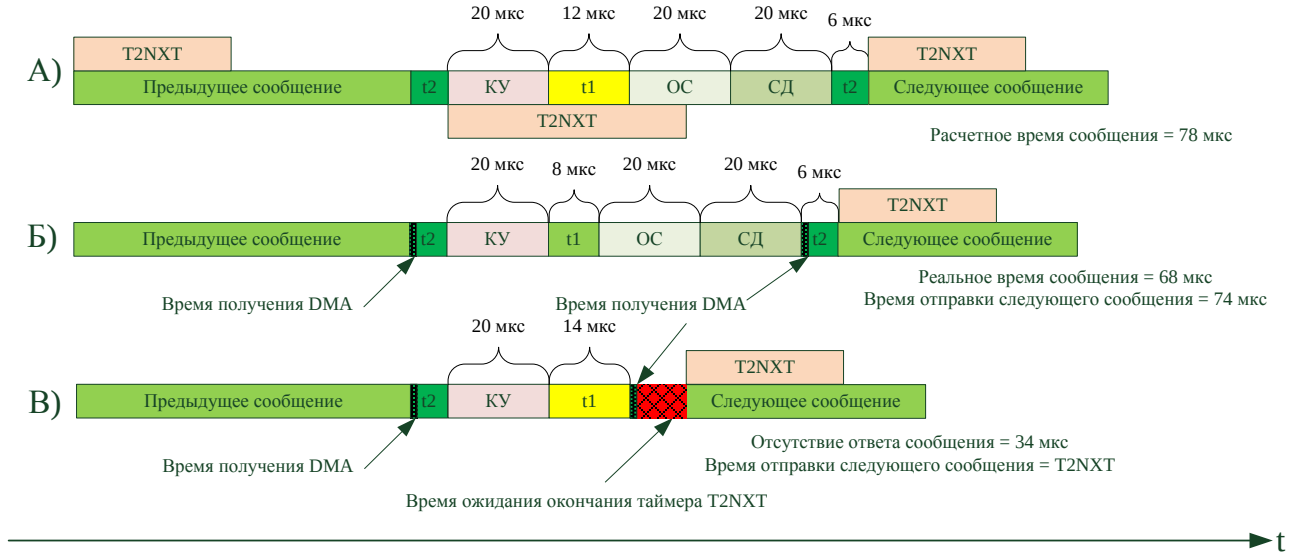

Рисунок 16. Использование таймера T2NXT со значением меньше длины сообщения.

На рисунке 16 представлен принцип работы таймера T2NXT с заданным значением меньше суммарного расчетного. В данном случае появляется возможность сократить суммарное время передачи сообщений в линии, обеспечив в то же время отсутствие встречного включения **КШ** и **ОУ** в случае запаздывания ответа на время больше **t1**. Хронометраж сети в случае появления ошибок будет достаточно непредсказуемым. При использовании функций кадрового и основного таймеров появляется возможность использовать “освободившееся время” до конца цикла или подцикла для передачи дополнительных (необязательных) сообщений. Остальные возможности привязки ко времени, как и в первом варианте. При отсутствии ответа от абонента или ОС с ошибкой без СД (если время T2NXT перекрывало СД) так же возможен анализ третьего слова DMA **КШ** со значением T2NXT.

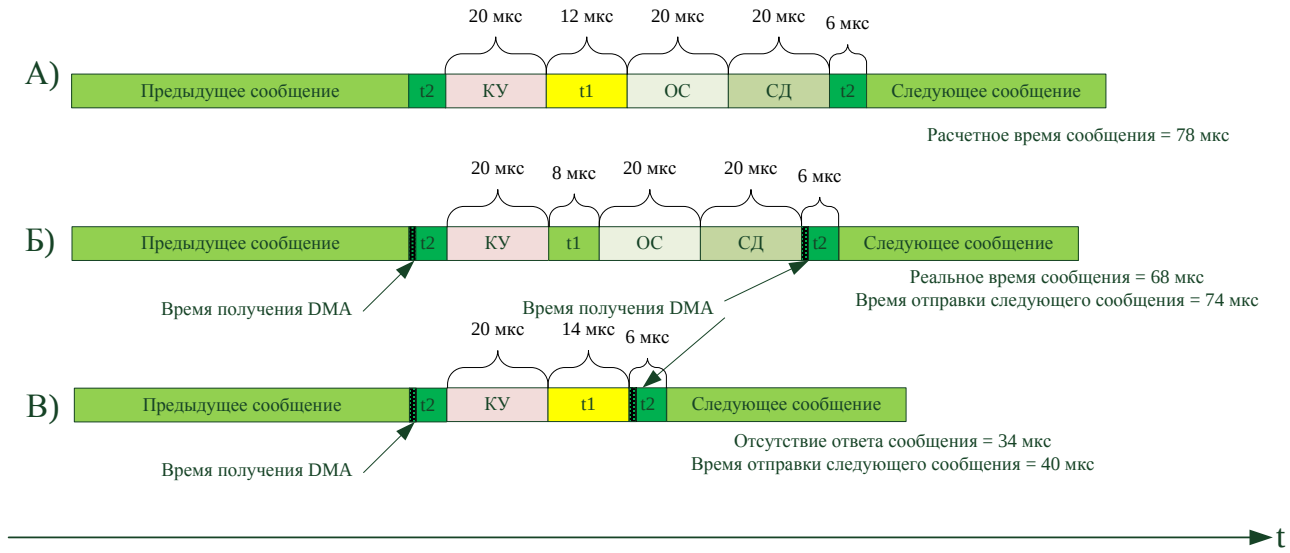

Рисунок 17. Сообщения без использования таймера T2NXT.

На рисунке 17 представлена последовательность сообщений без использования таймера T2NXT. В данном случае появляется возможность сократить суммарное время передачи сообщений в линии но, не обеспечивая в то же время, отсутствие встречного включения **КШ** и **ОУ** в случае запаздывания ответа на время больше **t1**. Хронометраж сети в случае появления ошибок будет также непредсказуемым, как и во втором варианте. При использовании функций кадрового и основного таймеров появляется возможность использовать “освободившееся время” до конца цикла или подцикла для передачи дополнительных (необязательных) сообщений. Остальные возможности привязки ко времени, как и в первом варианте.

Если необходимо гарантированно обеспечить отправку сообщений **КШ** в строго определенное время, используется первый вариант. Варианты 2 и 3 этой возможности не гарантируют.

Для корректной установки основного таймера СВСТ рекомендуется два варианта использования инструкций:

Вариант использования инструкции LTH с полем параметра = 0x0000 без предварительной инструкции LTT сразу после инструкции XEQ с КУ “Синхронизация”. В этом случае таймер СВСТ будет сброшен сразу после окончания КУ в КШ, так же как и в ОУ.

Вариант использования пары инструкций LTT и LTH после инструкции XEQ с КУ “Синхронизация”. В этом случае в полях параметров этих инструкций, можно указывать задержку в линии (или другие параметры для компенсации задержек) для более точной синхронизации таймеров КШ и ОУ.

Продолжение следует.

12. Список исправлений и изменений.

01.11.2019	1.0	Релиз документа.
11.02.2020	1.1	1. Исправлены значения временных задержек автомата КШ. Распределение памяти КШ и работа модулей "PCIe-1553UDx", "XMC-1553UDx", "CPCIS-1553UDx", "mPCIe-1553UDx" в режиме КШ. 2. Добавлен п. 8. Пример обработки КУ0 «Принять управление интерфейсом» в режиме КШ. Пример обработки КУ0 «Принять управление интерфейсом» в режиме КШ. 3. Добавлен п. 9. Пример обработки КУ0 «Принять управление интерфейсом» в режимах ОУ и АМШ. Пример обработки КУ0 «Принять управление интерфейсом» в режимах ОУ и АМШ.
03.03.2020	1.2	1. Добавлена иллюстрация превышение интервала t1 для примера 2 главы Системная область данных DMA. 2. Добавлен п. 10. Организация и работа таймеров. Организация и работа таймеров.
06.03.2020	1.3	1. Добавлен п. 11. Связь таймера T2NXT с циклами DMA, а так же основным и абсолютным таймерами. Связь таймера T2NXT с циклами DMA, а так же основным и абсолютным таймерами.
19.03.2020	1.4	Отредактирован и доработан п. 11
04.02.2021	1.5	Отредактирован и доработан п. 9
09.06.2023	1.6	Исправлена нумерация рисунков. Добавлена поддержка модулей LAN-MIL1553UDx и USB-MIL1553UDx.