Глава 9. SLAVE FIFOs

9.1 Введение

Хотя некоторые основанные на FX2 устройства могут использовать микропроцессор FX2, чтобы обрабатывать данные USB непосредственно (смотри Главу 8 "Доступ к Буферам Конечной точки"), большинство использует FX2 просто как канал между USB и внешней логикой, обрабатывающей данные (например, ASIC* или DSP, или контроллер IDE на жестком дисководе).

В устройствах с внешней логикой обработки данных данные USB текут между хостом и той самой внешней логикой - *обычно без какого-либо участия микропроцессора FX2* - через внутренне буферы *конечной точки* FX2. Для внешней логики эти буферы FIFO конечной точки выглядят похожими на большинство других; они обеспечивают обычную синхронизацию сигналов, квитируют связи строки (полный, пустой, программируемый уровень), стробы записи и чтения, разрешение выходного сигнала и т.п.

Эти FIFO сигналы должны, конечно же, управляться FIFO «мастером». Универсальный Программируемый Интерфейс (GPIF) FX2 может действовать как внутренний мастер, когда FX2 подключен к внешней логике, которая не включает стандартный интерфейс FIFO, или буферы FIFO могут управляться внешним мастером. Пока буферы FIFO управляются внешним мастером, FX2 будет в режиме **«Slave FIFO».**

Глава 10, " Универсальный Программируемый Интерфейс (GPIF)," описывает внутренний GPIF мастер-интерфейс. Настоящая же глава дает подробное описание интерфейса - как аппаратные средства, так и программное обеспечение - между подчиненными буферами FIFO FX2 и внешним мастером.

**** ASIC - (Application-Specific Integrated Circuit) проблемно-ориентированная (специализированная) интегральная микросхема, заказная ИС.***

Перевод сделан Потапенко О.Д., г. Ростов-на-Дону,
и выложен на сайте Д.С. Иоффе
http://www.dsioffe.narod.ru/,
любезно им предоставленном

октябрь 2005 г.

9.2 Аппаратное обеспечение

Рис. 9-1 иллюстрирует четыре буфера **Slave FIFO.** Рисунок показывает буферы FIFO, работающие в 16-разрядном режиме, хотя они также могут быть сконфигурированы для работы в 8-разрядном режиме.

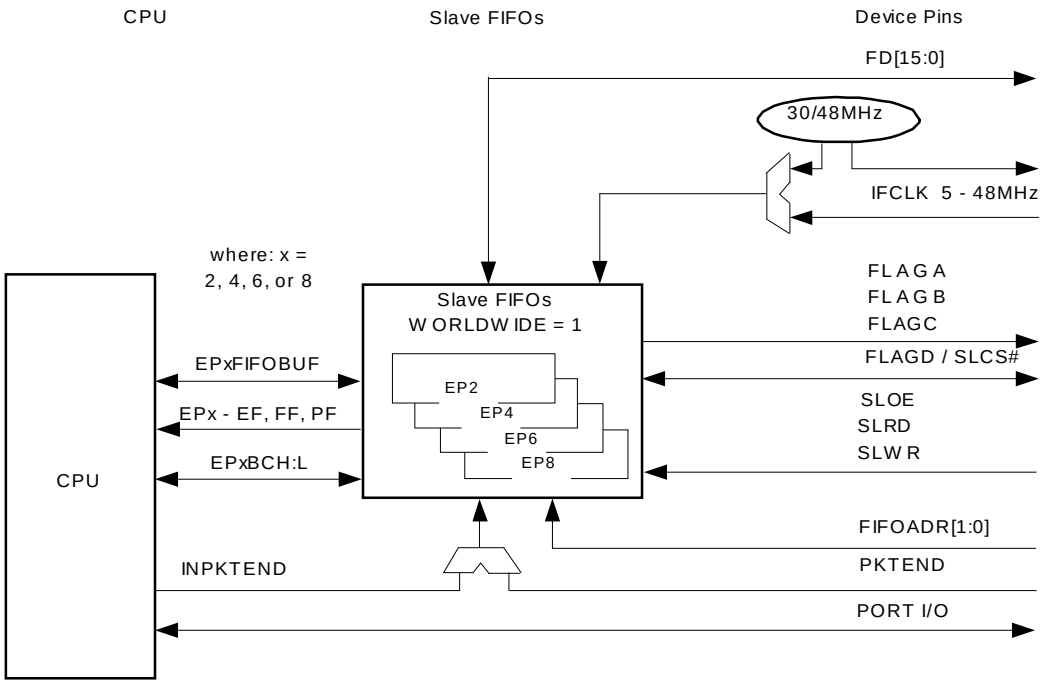

Рисунок 9-1. Роль Slave FIFO в системе FX2

В Таблице 9-1 перечислены регистры, связанные с аппаратными средствами **Slave FIFO.** Регистры полностью описаны в Главе 15, "Регистры".

Таблица 9-1. Регистры, относящиеся к FIFO

IFCONFIG	EPxFIFOPFH/L
PINFLAGAB	PORTACFG
PINFLAGCD	INPKTEND
FIFORESET	EPxFLAGIE
FIFOPINPOLAR	EPxFLAGIRQ
EPxCFG	EPxFIFOBCH:L
EPxFIFOCFG	EPxFLAGS
EPxAUTOINLENH:L	EPxBUF

9.2.1 Выводы SLAVE FIFO

При выходе FX2 из состояния СБРОС его выводы **I/O** сконфигурированы в режиме **PORTS**, а не в режиме **Slave FIFO**. Для того чтобы сконфигурировать выводы в режим **Slave FIFO**, биты **IFCFG[1:0]** в регистре **IFCONFIG** должны быть установлены в **'11'** (см. Табл. 13-10, "Выбор **IFCFG** функций выводов порта ввода/ввода" более детально).

Когда **IFCFG[1:0] = 11**, интерфейсные выводы SLAVE FIFO представлены внешнему мастеру, как показано на Рис. 9-2.

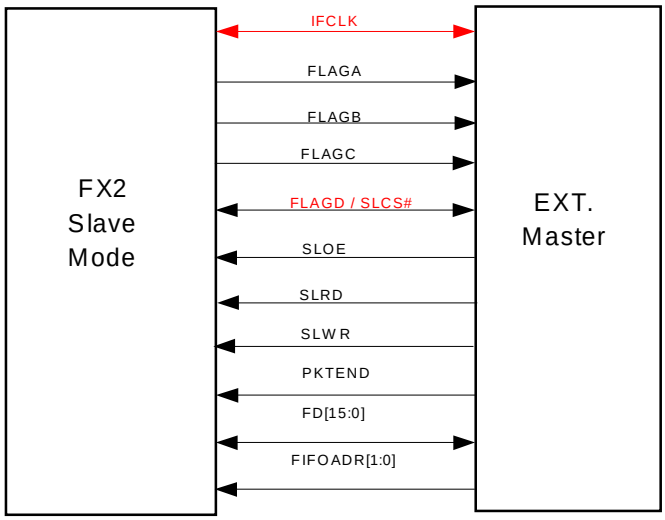

Рис. 9-2. Полнофункциональные интерфейсные выводы FX2 в режиме Slave

Внешняя логика имеет доступ к буферам FIFO через 8- или 16-разрядную шину данных FD. Шина данных двунаправленная, с выходными драйверами, управляемыми выводом **SLOE.**

Выводы FIFOADR[1:0] выбирают, которое из четырех FIFO подключено к шине FD.
В асинхронном режиме **(IFCONFIG.3 = 1) SLRD** и **SLWR** – стробы чтения и записи соответственно.
В синхронном режиме **(IFCONFIG.3 = 0) SLRD** и **SLWR** тактируются по выводу **IFCLK.**

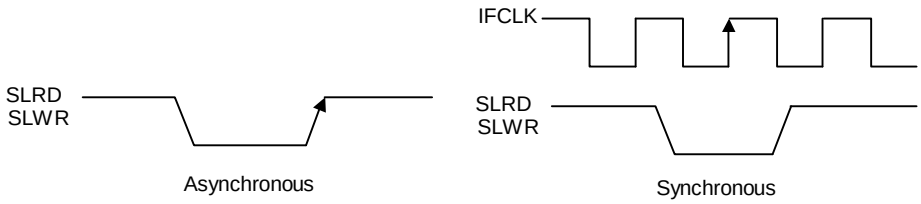

Рисунок 9-3. Асинхронная и синхронная модели с учетом задержек

9.2.2 Шина данных FIFO (FD)

Шина данных FIFO, **FD[x:0]**, может быть 8- или 16-разрядной. Разрядность шины выбирается с помощью бита **WORDWIDE, (EPxFIFOCFG.0)** для каждого FIFO:

- **WORDWIDE=0**: 8-разрядный режим. **FD[7:0]** заменяет **Port B** (Смотри Рис. 9-4)
- **WORDWIDE=1**: 16-разрядный режим. **FD[7:0]** заменяет **Port B** и **FD[15:8]** заменяет **Port D** (Смотри Рис. 9-5)

При СБРОСе по включению питания, шина данных FIFO устанавливается по умолчанию в 16-разрядный режим **(WORDWIDE = 1)** для всех FIFO.

В любом режиме выводы **FIFOADR[1:0]** выбирают, которое из четырех FIFO непосредственно подключены к выводам **FD**.

*Если все FIFO сконфигурированы в 8-разрядном режиме, Port D остается доступным для использования как порт ввода/вывода общего назначения. Если любое FIFO сконфигурировано для работы в 16-разрядном режиме, Port D запрещен для использования как порт ввода/вывода общего назначения, невзирая на то, который FIFO в настоящее время выбран с помощью выводов **FIFOADR[1:0]**.*

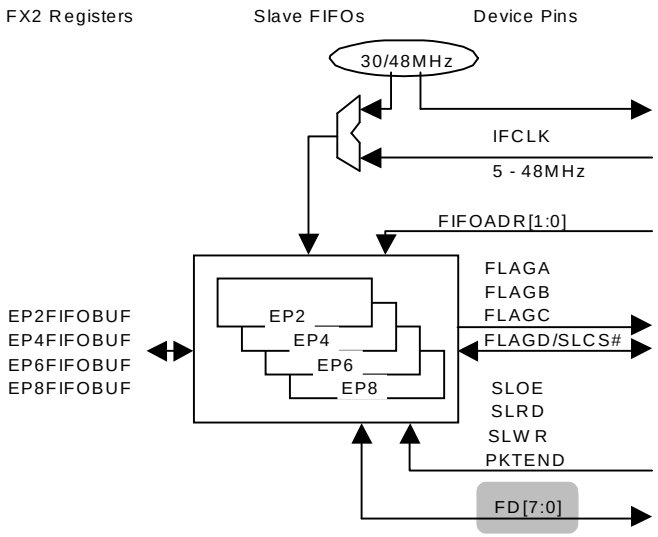

Рис. 9-4. 8-разрядный режим Slave FIFO, WORDWIDE=0

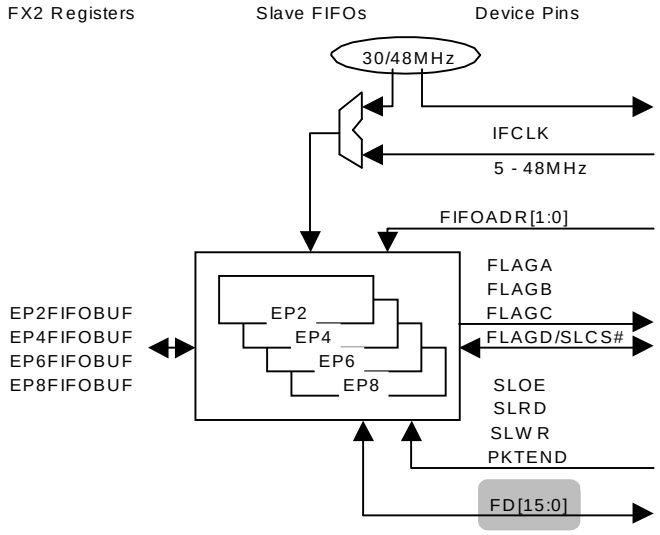

Рис. 9-5. 16-разрядный режим работы буферов Slave FIFO, WORDWIDE=1

9.2.3 Синхронизация Интерфейса (IFCLK)

Интерфейс **Slave FIFO** может тактироваться внутренним или внешним источником тактовых сигналов. Внутренний источник тактов FX2 может быть сконфигурирован для работы с частотой 30 или 48 МГц и может быть выведен дополнительно на вывод **IFCLK**. Если FX2 сконфигурирован для использования внешнего источника тактов, вывод **IFCLK** может управляться сигналами с любой частотой от 5 МГЦ до 48 МГЦ. При СБРОСе по включению питания FX2 устанавливается по умолчанию в режим работы от внутреннего источника тактов 48 МГЦ, нормальная полярность, с запретом выхода **IFCLK**. Смотри Рис. 9-6.

Бит **IFCONFIG.7** выбирает внутренний или внешний источник тактов:
'0' = внешний, **'1'** = внутренний.

Бит **IFCONFIG.6** выбирает частоту внутреннего источника тактов 30 или 48 МГц:
'0' = 30 МГЦ, **'1'** = 48 МГЦ. Этот бит не имеет значения при **IFCONFIG.7 = 0**.

Бит **IFCONFIG.5** разрешает выходной сигнал внутреннего источника тактов:
'0' = запрет выхода, **'1'** = разрешение. Этот бит не имеет значения при **IFCONFIG.7 = 0**.

Бит **IFCONFIG.4** инвертирует полярность тактовых сигналов интерфейса (независимо от того, является ли источник тактов внутренним или внешним): 0 = нормально, 1 = инверсия. Инверсия **IFCLK** может облегчить сопряжение FX2 с определенного рода внешней схемой.

Рис. 9-7, например, демонстрирует использование инверсии **IFCLK** для того, чтобы гарантировать достаточно длительное время установки для чтения флагов FIFO FX2.

*Когда **IFCLK** сконфигурирован как вход, минимальная частота, которая может быть приложена к этому выводу - 5 МГЦ.*

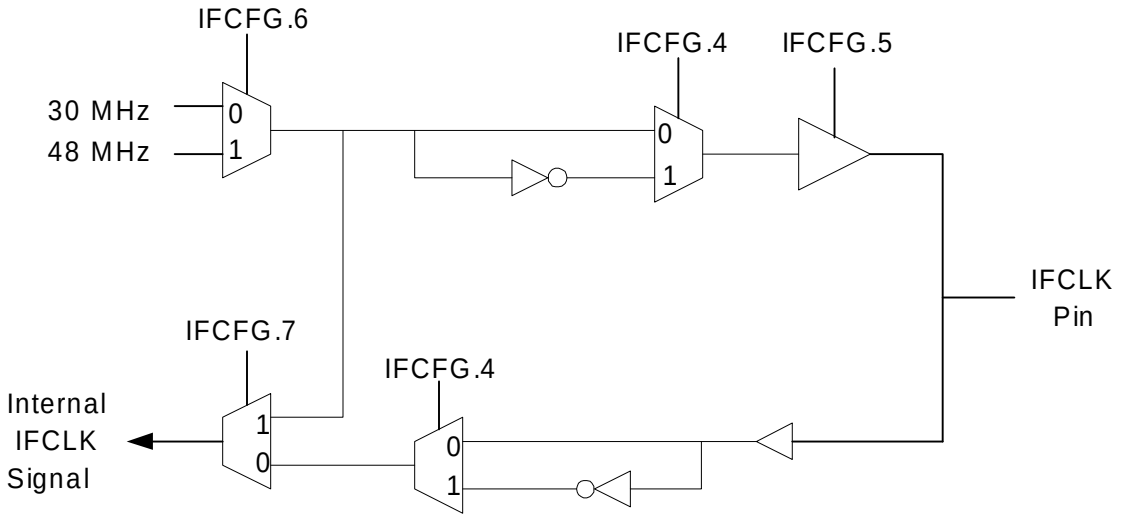

Рис. 9-6. Конфигурация IFCLK

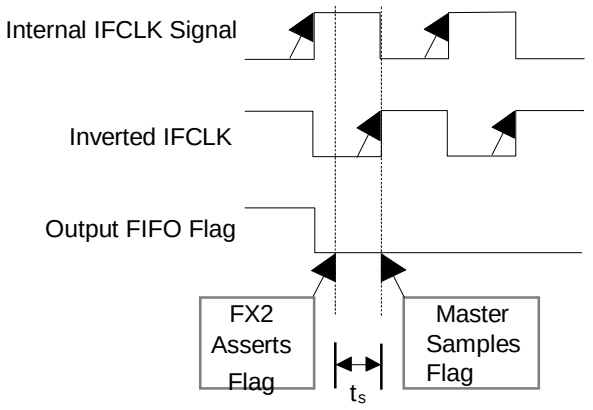

Рис. 9-7. Удовлетворение времени установки инверсией выхода IFCLK

9.2.4 Выводы флага FIFO (FLAGA, FLAGB, FLAGC, FLAGD)

Четыре вывода **– FLAGA, FLAGB, FLAGC** и **FLAGD –** сообщают статус буферов FIFO FX2. В дополнение к обычным сигналам «FIFO заполнен» и «FIFO пуст» есть также сигнал, который указывает, что FIFO заполнился до уровня, запрограммированного пользователем.

Внешний мастер обычно проверяет флаг «пусто» выходных конечных точек (**OUT EPs**) и флаг «заполнено» входных конечных точек (**IN EPs**).

Флаг «программируемый уровень» одинаково полезный для любого типа конечной точки (это может, например, дать заблаговременное предупреждение, что конечная точка **OUT** почти пустая или что конечная точка **IN** почти полная).

Выводы **FLAGA, FLAGB** и **FLAGC** могут оперировать в любом из двух режимов: *Индексном* или *Фиксированном*, выбранном с помощью регистров **PINFLAGSAB** и **PINFLAGSCD**. Вывод **FLAGD** действует только в *Фиксированном* режиме. Каждый вывод конфигурируется независимо; некоторые выводы могут быть в *Фиксированном* режиме, тогда как другие - в *Индексном*. Смотри Главу 15, "Регистры" для подробной информации.

Выводы флагов, сконфигурированные для *Индексного* режима, сообщают статус FIFO, выбранного с помощью выводов **FIFOADR[1:0]** к настоящему времени. В *Индексном* режиме **FLAGA** сообщает состояние «программируемого уровня», **FLAGB** сообщает статус «заполнено» и **FLAGC** сообщает статус «пусто».

Выводы Флагов, сконфигурированные для *Фиксированного* режима, сообщают одно из трех условий для специфического FIFO, независимо от состояния выводов **FIFOADR[1:0]**. Условие и FIFO выбираются пользователем. Например, **FLAGA** мог бы быть сконфигурирован, чтобы сообщать статус «пусто» буфера FIFO2, **FLAGB** - чтобы сообщать статус «пусто» буфера FIFO4, **FLAGC** - чтобы сообщать статус «программируемого уровня» FIFO4 и **FLAGD** - чтобы сообщать статус «заполнено» FIFO6.

Полярность выводов флагов «пусто» и «заполнено» устанавливается по умолчанию в активно-низкий уровень, но может быть инвертирована с помощью регистра **FIFOPINPOLAR**.

При СБРОСе по включению питания флаги FIFO сконфигурированы для *Индексной* операции.

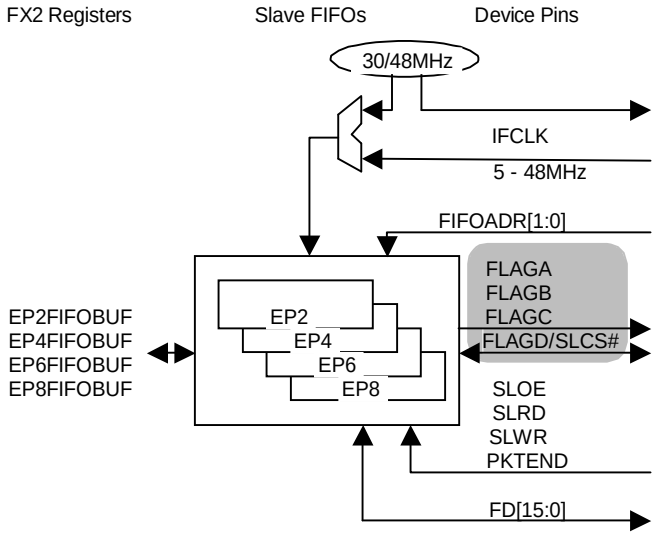

Рис. 9-8. ФЛАГх (FLAGx)

9.2.5 Управляющие выводы (SLOE, SLRD, SLWR, PKTEND, FIFOADR[1:0])

«Управляющими» (“control”) выводами **Slave FIFO** являются:

- **SLOE (Slave Output Enable)** - разрешение выхода
- **SLRD (Slave Read)** - чтение
- **SLWR (Slave Write)** - запись
- **PKTEND (Packet End)** - конец пакета
- **FIFOADR[1:0] (FIFO Select)** - выбор FIFO

“Read” и **“Write”** считаются относительно внешнего мастера; внешний мастер читает из **OUT EP** и пишет в **IN EP.** Смотри Рис. 9-9.

Чтение — SLOE and SLRD:

Вывод **SLOE** разрешает **FD** выходы.

По умолчанию **SLOE** и **SLRD** - активно-низкого уровня; их полярности могут быть изменены через регистр **FIFOPINPOLAR.**

В синхронном режиме **(IFCONFIG.3 = 0)** указатель FIFO увеличивается по каждому переднему фронту сигнала **IFCLK,** пока **SLRD** активен. В асинхронном режиме **(IFCONFIG.3 = 1)** указатель FIFO увеличивается по каждому переходу **SLRD** из активного состояния в неактивное.

Запись — SLWR:

По умолчанию, **SLWR** - активно-низкого уровня; его полярность может быть изменена через регистр **FIFOPINPOLAR.**

В синхронном режиме **(IFCONFIG.3 = 0)** данные на шине **FD** записываются в FIFO (и указатель FIFO увеличивается) по каждому переднему фронту **IFCLK,** пока **SLWR** активен. В асинхронном режиме **(IFCONFIG.3 = 1)** данные на шине **FD** записываются в FIFO (и указатель FIFO увеличивается) по каждому переходу **SLRD** из активного состояния в неактивное.

FIFOADR[1:0]:

Выводы **FIFOADR[1:0]** выбирают, которое из четырех FIFO подключено к шине **FD** (и, если флаги FIFO действуют в *Индексном* режиме, выбирают, какие флаги FIFO представлены на выводах **FLAGx**):

Таблица 9-2. Выбор FIFO посредством FIFOADR[1:0]

FIFOADR[1:0]	Selected FIFO
00	EP2
01	EP4
10	EP6
11	EP8

PKTEND:

Внешний мастер утверждает вывод **PKTEND**, чтобы совершить **IN** пакет на USB независимо от длины пакета. **PKTEND** обычно используется, когда мастер хочет послать «короткий» пакет (то есть пакет, меньший, чем размер, определенный в регистрах **EPxAUTOINLENH:L**).

Например:

Допустим, что **EP4AUTOINLENH:L** установлен по умолчанию в состояние 512 байтов. Если **AUTOIN = 1**, внешний мастер может передавать данные в FIFO4 непрерывно, и (в отсутствии каких-либо узких мест на пути данных) FX2 автоматически будет передавать пакет на USB всякий раз, когда FIFO заполнится 512 байтами. Если мастер захочет послать поток данных, чья длина не является кратным 512, последний пакет не будет автоматически передан на USB, поскольку он меньше, чем 512 байтов. Чтобы передать этот последний пакет, мастер может сделать одно из двух: он может заполнить пакет с ложными данными для того, чтобы сделать длину пакета равной точно 512 байтам, или он может записать короткий пакет в FIFO, а затем утвердить вывод **PKTEND.**

Если FIFO сконфигурирован, чтобы разрешать пакеты нулевой длины **(EPxFIFOCFG.2 = 1)**, утверждение вывода **PKTEND**, когда FIFO «заполнен», вызовет передачу пакета нулевой длины.

По умолчанию, **PKTEND** - активно-низкого уровня; его полярность может быть изменена через регистр **FIFOPINPOLAR**.

*Вывод **PKTEND** не должен утверждаться, пока буфер не станет доступным, даже если передаваться должен только пакет нулевой длины. Флаг «заполнено» может быть использован, чтобы определять, является ли буфер доступным.*

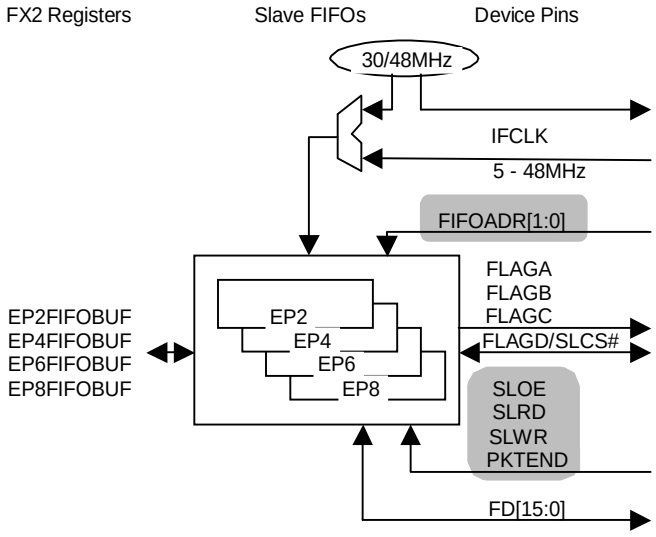

Рис. 9-9. Управляющие выводы Slave FIFO (Slave FIFO Control Pins)

9.2.6 Slave FIFO Chip Select (SLCS#)

Вывод **“Slave FIFO Chip Select” (SLCS#) –** альтернативная функция вывода **PA7**; он разрешается через бит **PORTACFG.6** (смотри Раздел 13.3.1, "Port A, Дополнительные Функции").

Вывод **SLCS#** позволяет внешней логике эффективно удалять FX2 с шины данных FIFO для того, чтобы, например, совместно использовать эту шину среди многочисленных ведомых устройств.

Пока на выводе **SLCS#** действует высокий уровень от внешней логики, FX2 поддерживает свои выводы **FD[x:0]** в третьем состоянии и игнорирует выводы **SLOE, SLRD, SLWR** и **PKTEND.**

9.2.7 Implementing Synchronous Slave FIFO Writes

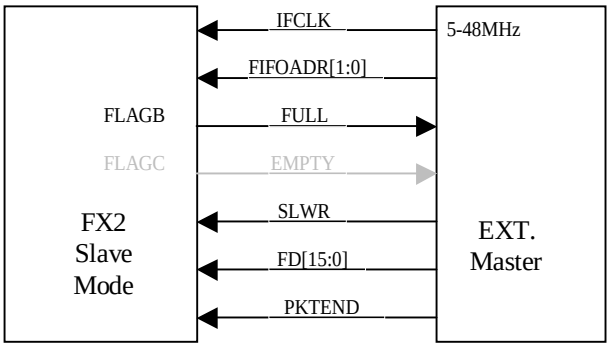

Рис. 9-10. Пример интерфейсных выводов: Синхронная ЗАПИСЬ в FIFO

Обычно, последовательность событий для внешнего мастера такова:

СОСТОЯНИЕ ОЖИДАНИЯ: Когда событие записи происходит, переход в Состояние 1.

СОСТОЯНИЕ 1: Указывается операция **IN FIFO**, утверждается **FIFOADR[1:0]**, переход в Состояние 2.

СОСТОЯНИЕ 2: Если флаг «FIFO заполнен» является ложью (FIFO не полон), переход в Состояние 3, иначе остается в Состоянии 2.

СОСТОЯНИЕ 3: Выставляются данные на шину, утверждается **SLWR** на один такт **IFCLK**, переход в Состояние 4.

СОСТОЯНИЕ 4: Если есть еще данные для записи, переход в Состояние 2, иначе переход в СОСТОЯНИЕ ОЖИДАНИЯ.

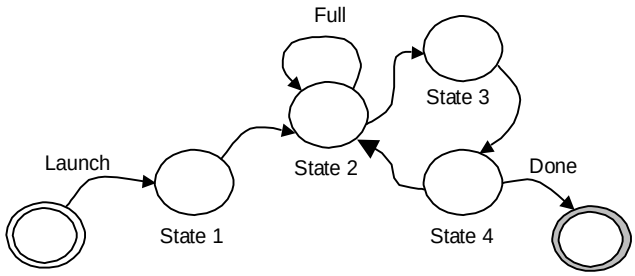

Рис. 9-11. Пример автомата состояний: Синхронная ЗАПИСЬ в FIFO

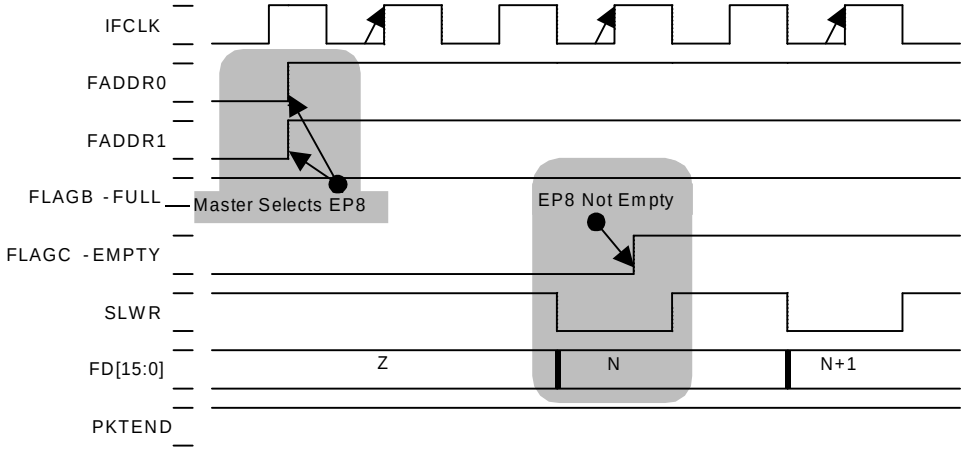

Рис. 9-12. Пример диаграммы: Синхронная ЗАПИСЬ в FIFO, Waveform 1

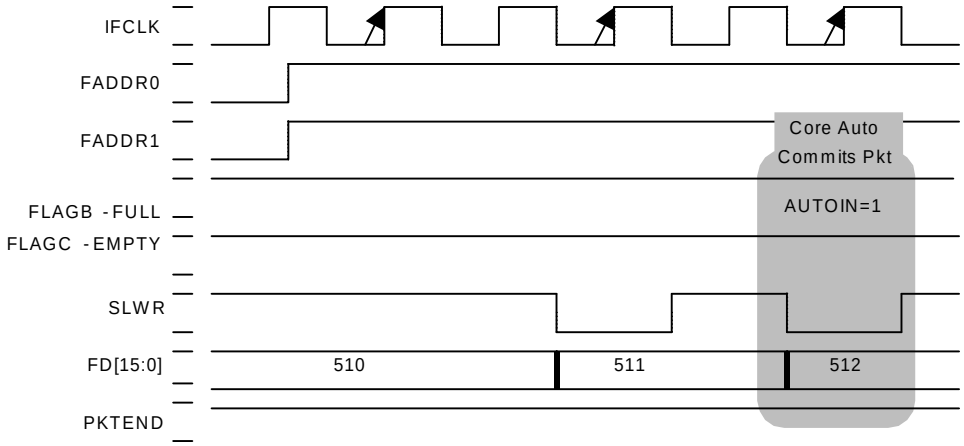

Рис. 9-13. Пример диаграммы: Синхронная ЗАПИСЬ в FIFO, Waveform 2

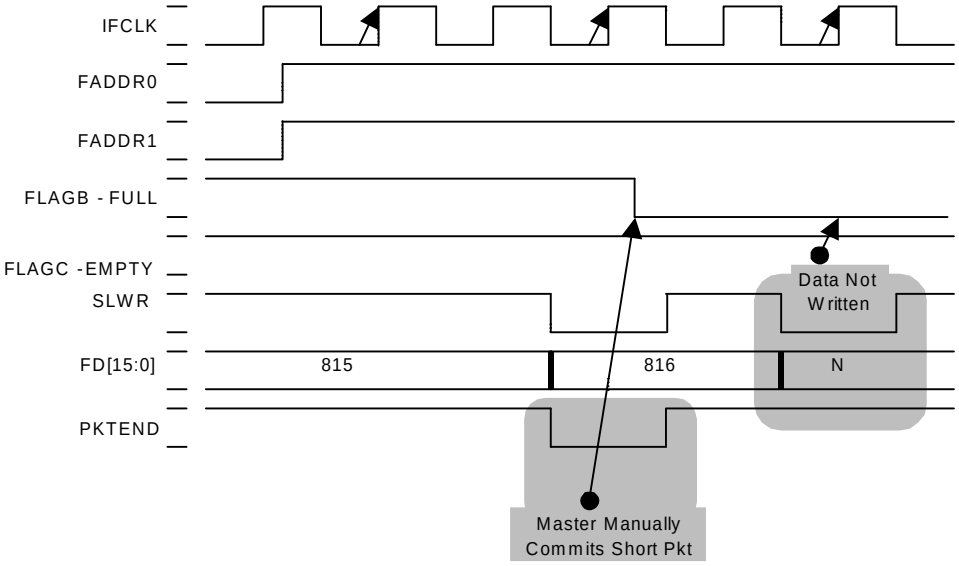

Рис. 9-14. Пример диаграммы: Синхронная ЗАПИСЬ в FIFO, Waveform 3, иллюстрация вывода PKTEND

9.2.8 Выполнение Синхронного ЧТЕНИЯ Slave FIFO (Slave FIFO Reads)

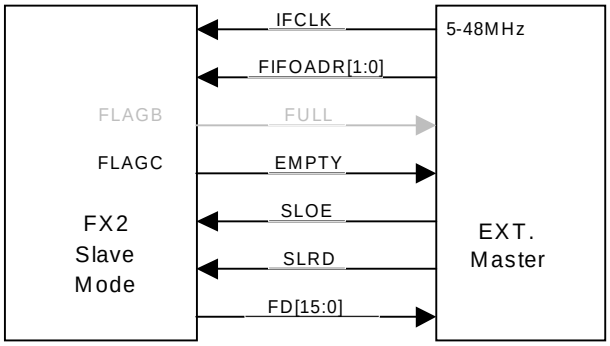

Рис. 9-15. Пример интерфейсных выводов: Синхронное ЧТЕНИЕ FIFO

Обычно, последовательность событий для внешнего мастера такова:

СОСТОЯНИЕ ОЖИДАНИЯ: Когда событие ЧТЕНИЕ происходит, переход в Состояние 1.

СОСТОЯНИЕ 1: Указывается операция **OUT FIFO,** утверждается **FIFOADR[1:0],** переход в Состояние 2.

СОСТОЯНИЕ 2: Утверждается **SLOE.** Если флаг «FIFO пуст» является ложным (FIFO не пуст), переход в Состояние 3, иначе остается в Состоянии 2.

СОСТОЯНИЕ 3: Выставляются данные на шину, инкрементируется указатель утверждением **SLRD** на один такт **IFCLK**, отрицается **SLOE,** переход в Состояние 4.

СОСТОЯНИЕ 4: Если есть еще данные для ЧТЕНИЯ, переход в Состояние 2, иначе переход в СОСТОЯНИЕ ОЖИДАНИЯ.

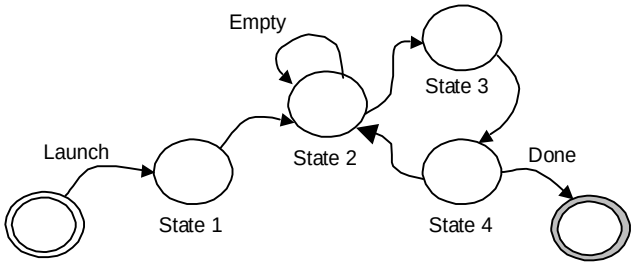

Рис. 9-16. Пример автомата состояний: Синхронное ЧТЕНИЕ FIFO

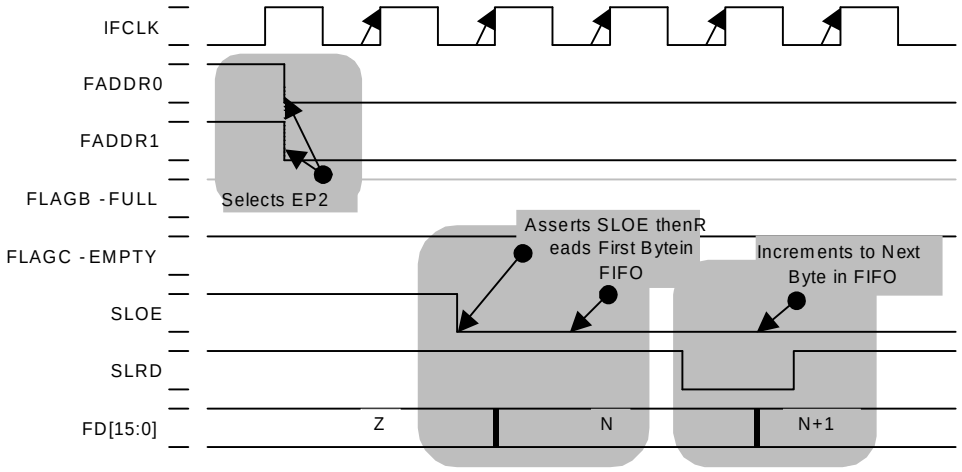

Рис. 9-17. Пример диаграмм: Синхронное ЧТЕНИЕ FIFO, Waveform 1

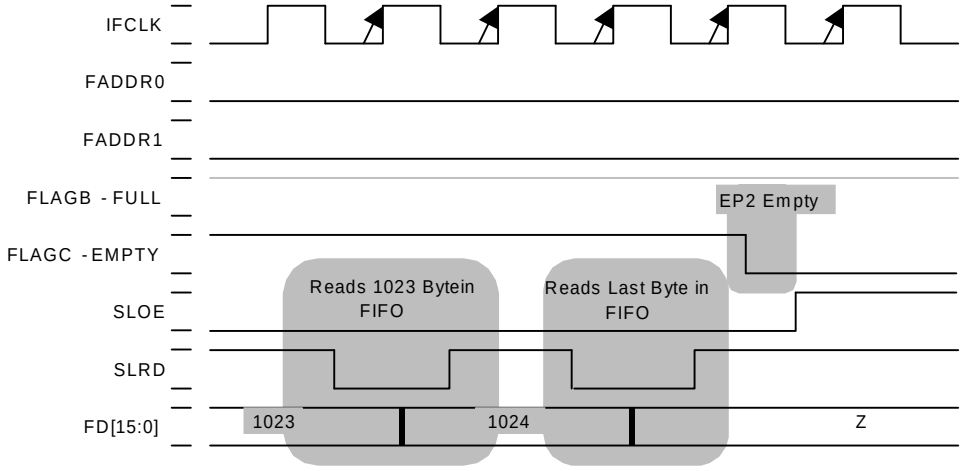

Рис. 9-18. Пример диаграмм: Синхронное ЧТЕНИЕ FIFO, Waveform 2, иллюстрация Флага «ПУСТО»

9.2.9 Выполнение Асинхронной Записи в Slave FIFO (Slave FIFO Writes)

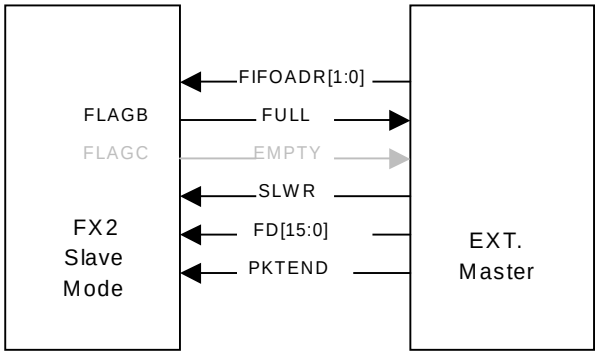

Рис. 9-19. Пример интерфейсных выводов: асинхронная запись в FIFO

Обычная последовательность событий для внешнего мастера:

СОСТОЯНИЕ ОЖИДАНИЯ: Когда событие записи происходит, переход в Состояние 1.

СОСТОЯНИЕ 1: Указывается операция **IN FIFO**, утверждается **FIFOADR[1:0]**, переход в Состояние 2.

СОСТОЯНИЕ 2: Если флаг «FIFO заполнен» является ложью (FIFO не полон), переход в Состояние 3, иначе остается в Состоянии 2.

СОСТОЯНИЕ 3: Выставляются данные на шину, инкрементируется указатель утверждением/отрицанием **SLWR,** переход в Состояние 4.

СОСТОЯНИЕ 4: Если есть еще данные для записи, переход в Состояние 2, иначе переход в СОСТОЯНИЕ ОЖИДАНИЯ.

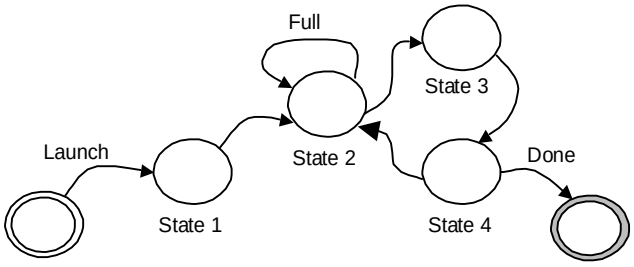

Рис. 9-20. Пример конечного автомата: Асинхронная ЗАПИСЬ в FIFO

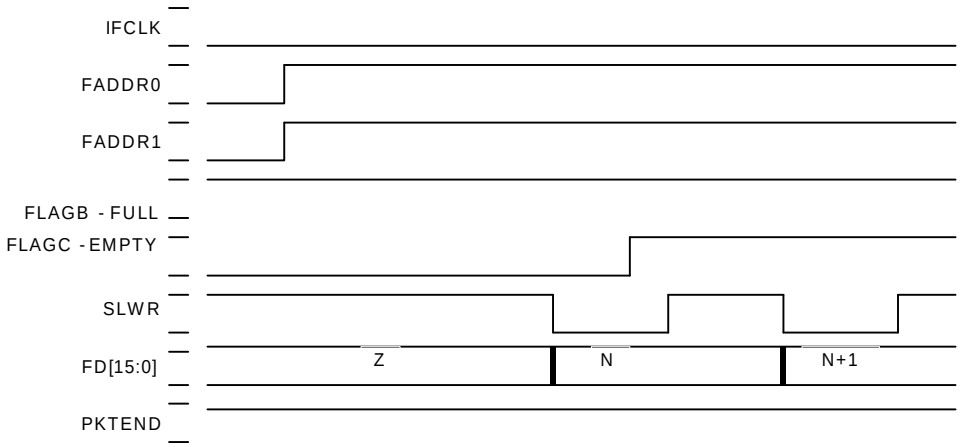

Рис. 9-21. Пример диаграмм: Асинхронная ЗАПИСЬ в FIFO

9.2.10 Выполнение Асинхронного ЧТЕНИЯ из Slave FIFO (Slave FIFO Reads)

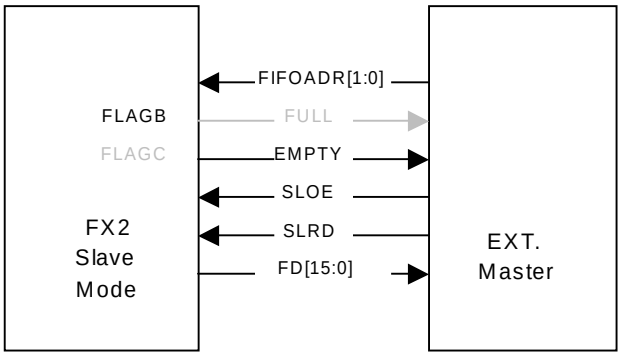

Рис. 9-22. Пример интерфейсных выводов: Асинхронное ЧТЕНИЕ из FIFO

Обычная последовательность событий для внешнего мастера:

СОСТОЯНИЕ ОЖИДАНИЯ **(IDLE):** Когда событие ЧТЕНИЕ происходит, переход в Состояние 1.

СОСТОЯНИЕ 1: Указывается операция **OUT FIFO**, утверждается **FIFOADR[1:0]**, переход в Состояние 2.

СОСТОЯНИЕ 2: Если флаг «FIFO пуст» является ложью (FIFO не пуст), переход в Состояние 3, иначе остается в Состоянии 2.

СОСТОЯНИЕ 3: Утверждается **SLOE,** утверждается **SLRD,** выставляются данные на шину, отрицается **SLRD (**инкремент указателя), отрицается **SLOE,** переход в Состояние 4.

СОСТОЯНИЕ 4: Если есть еще данные для ЧТЕНИЯ, переход в Состояние 2, иначе переход в СОСТОЯНИЕ ОЖИДАНИЯ.

.

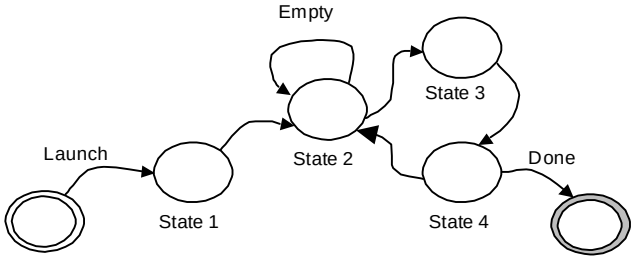

Рис. 9-23. Пример Конечного Автомата: Асинхронное ЧТЕНИЕ из FIFO

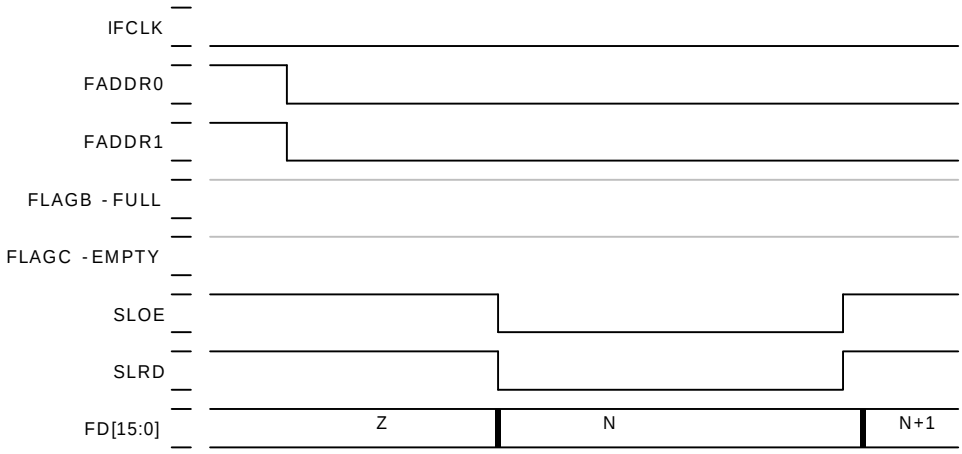

Рис. 9-24. Пример Диаграмм: Асинхронное ЧТЕНИЕ из FIFO

9.3 Микропрограммное управление (Firmware)

Этот раздел описывает интерфейс между микропрограммой FX2 и буферами FIFO. Больше информации доступно в Главе 8, "Доступ к Буферам Конечной точки".

Таблица 9-3. Регистры, связанные с микропрограммой Slave FIFO

EPxCFG	INPKTEND
EPxFIFOCFG	EPxFIFOIE
EPxAUTOINLENH/L	EPxFIFOIRQ
EPxFIFOPFH:L	INT2IVEC
EP2468STAT	INT4IVEC
EP24FIFOFLGS	INTSETUP
EP68FIFOFLGS	IE
EPxCS	IP EPx-
FIFOFLGS	INT2CLR
EPxBCH:L	INT4CLR
EPxFIFOBCH:L	EIE EPx-
FIFOBUF	EXIF
REVCTL (bits 0 and 1 must be initialized to 1 for operation as described in this chapter)	

9.3.1 Микропрограммный доступ к FIFO (Firmware FIFO Access)

Микропрограммы FX2 могут иметь доступ к **Slave FIFO**, используя четыре регистра в памяти **XDATA: EP2FIFOBUF, EP4FIFOBUF, EP6FIFOBUF** и **EP8FIFOBUF.** Эти регистры могут быть непосредственно прочитаны и записаны (используя инструкции **MOVX**) или они могут служить в качестве источников и адресатов информации для двойного механизма автоуказателя, построенного в EZ-USB FX2 (смотри Раздел 8.8 "Автоуказатели" (Autopointers)).

К тому же, есть ряд управляющих и статусных регистров FIFO: регистры Подсчета байтов показывают количество байтов в каждом FIFO; биты флагов указывают заполнение FIFO; биты режима управляют различными режимами FIFO и т.п.

Эта глава фокусирует внимание на регистрах и битах, которые характерны для операций со **Slave FIFO**; для полного описания всех регистров FIFO смотри Главу 8, "Доступ к Буферам Конечной точки" и Главу 15, "Регистры".

*Для правильной работы, как описано в этой главе, микропрограмма FX2 должна установить биты **DYN_OUT** и **ENH_PKT (REVCTL.0 и REVCTL.1) в '1'.***

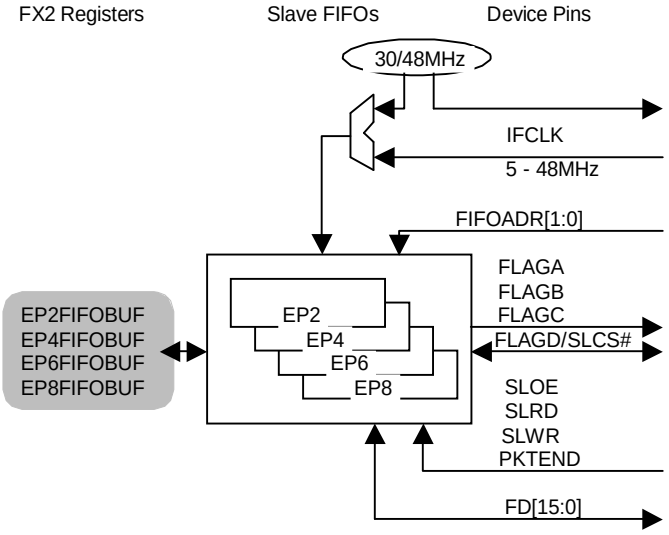

Рис. 9-25. Регистры EPxFIFOBUF

9.3.2 Память конечных точек EPx (EPx Memories)

Буферы **Slave FIFO** подключают внешнюю логику к четырем блокам памяти конечных точек FX2 (EP2, EP4, EP6 и EP8).

Эти блоки памяти имеют следующие программируемые характеристики:

1. Тип в соответствии с типом передачи может быть
   - Bulk – передача массива данных
   - INTERRUPT – передача по прерываниям
   - ISOCHRONOUS – изохронная передача
2. Направление может быть: **“В”** или **“ИЗ” (IN** или **OUT).**
3. Размер может быть: для EP2 и EP6 - 512 или 1024 байтов;
   для EP4 и EP8 - только 512 байтов.
4. Буферизация может быть: для EP2 и EP6 - 2-х, 3-х или 4-х кратная;
   для EP4 и EP8 - только 2-х кратная.
5. FX2 автоматически вводит данные конечной точки в/из интерфейса **Slave FIFO (AUTOIN=1, AUTOOUT=1).**

При СБРОСе по включению питания эти блоки памяти конечных точек конфигурируются следующим образом:

1. EP2 - Bulk OUT, 512 байтов/пакет, 2-х кратная буферизация
2. EP4 - Bulk OUT, 512 байтов/пакет, 2-х кратная буферизация
3. EP6 - Bulk IN, 512 байтов/пакет, 2-х кратная буферизация
4. EP8 - Bulk IN, 512 байтов/пакет, 2-х кратная буферизация

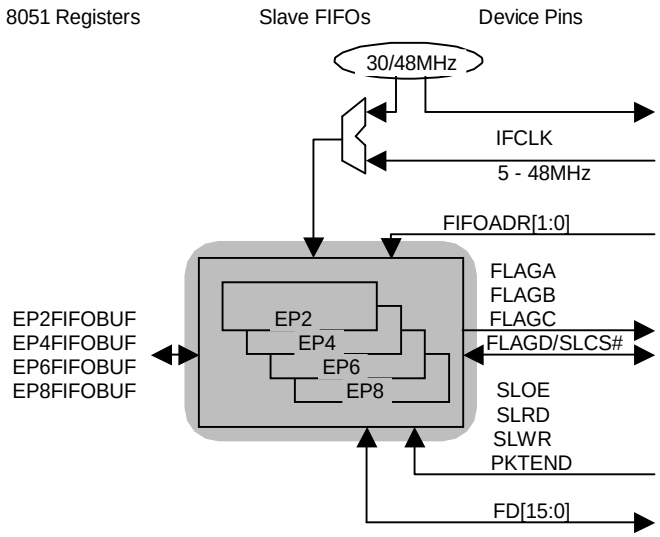

Рис. 9-26. Блоки памяти EPx

9.3.3 Флаг программируемого уровня буфера Slave FIFO Programmable-Level Flag (PF)

Каждый флаг программируемого уровня (PF) буферов FIFO утверждается, когда FIFO достигает порога заполнения, определенного пользователем. Этот порог конфигурируется следующим образом:

1. Для пакетов **OUT:** порог запоминается в **PFC[12:0].**
   PF утверждается, когда количество байтов *в целом FIFO* **меньше или равно (при DECIS=0)** или **больше или равно (при DECIS=1)** порога.
2. Для **IN** пакетов с **PKTSTAT = 1:** порог запоминается в **PFC[9:0].**
   PF утверждается, когда количество байтов, записанных в текущем пакете в FIFO, меньше или равно **(при DECIS=0)** или больше или равно **(при DECIS=1)** порога.
3. Для **IN** пакетов с **PKTSTAT = 0:** порог запоминается в двух частях: **PKTS[2:0]** хранит количество переданных пакетов, а **PFC[9:0]** хранит количество байтов в текущем пакете.
   PF утверждается, когда буфер FIFO либо менее полон **(при DECIS=0),** либо более полон **(при DECIS=1),** чем установленный порог.

По умолчанию, **FLAGA** - флаг программируемого уровня **(PF)** для конечной точки, определенной в настоящий момент времени выводами **FIFOADR[1:0].**

По умолчанию, для **EP2** и **EP4** конфигурация конечной точки - **BULK, OUT, 512, 2-x** кратная буферизация и вывод **PF** утверждается, когда объем всего FIFO больше или равен 512 байтам.

По умолчанию, для **EP6** и **EP8** конфигурация конечной точки - **BULK, IN, 512, 2-x** кратная буферизация и вывод **PF** утверждается, когда объем всего FIFO менее или равен 512 байтам.

Другими словами, сконфигурированные по умолчанию **PF** утверждаются:

- для EP2 и EP4, когда буферы FIFO наполовину полные,
- для EP6 и EP8, когда буферы FIFO наполовину пустые.

Смотри Главу 15, "Регистры" для подробного изложения.

9.3.4 Режимы Auto-In / Auto-Out

Буферы FIFO FX2 могут быть сконфигурированы так, чтобы передавать пакеты в/из USB автоматически. Для **IN-**конечных точек режим **Auto-In** позволяет внешней логике передавать данные в FIFO непрерывно, без какой либо для неё (логики) или FX2 потребности в микропрограмме для выполнения пакетирования данных или явного предупреждения FX2 о необходимости посылать данные в хост-контроллер. Для **OUT-**конечных точек режим **Auto-Out** позволяет хосту непрерывно заполнять FIFO без какой-либо потребности во внешней логике или микропрограммах FX2 для подтверждения получения каждого поступающего пакета, активирования буферов конечных точек и т.п. Смотри **Рис. 9-27.**

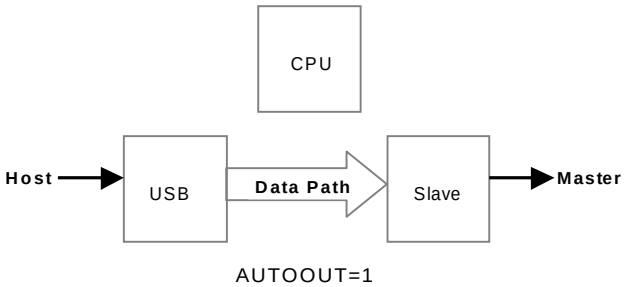

Рис. 9-27. Когда AUTOOUT=1, OUT-пакеты передаются автоматически

Для того чтобы сконфигурировать **IN**-конечную точку FIFO для режима **Auto (Auto Mode),** установите бит **AUTOIN** в соответствующем регистре **EPxFIFOCFG** в **'1'**.

Для того чтобы сконфигурировать **OUT**-конечную точку FIFO для режима **Auto (Auto Mode),** установите бит **AUTOOUT** в соответствующем регистре **EPxFIFOCFG** в **'1'**.

Смотри **Рис. 9-28** и **Рис. 9-29.**

При *СБРОС*е по включению питания все буферы FIFO по умолчанию устанавливаются в *Ручной Режим* (т.е. **AUTOIN = 0** и **AUTOOUT = 0**).

```
TD_Init():
… … … … …
REVCTL = 0x03;       // MUST set REVCTL.0 and REVCTL.1 to 1
SYNCDELAY;
EP2CFG = 0xA2;       // EP2 is DIR=OUT, TYPE=BULK, SIZE=512, BUF=2x
SYNCDELAY;
FIFORESET = 0x80;    // Reset the FIFO
SYNCDELAY;
FIFORESET = 0x02;
SYNCDELAY;
FIFORESET = 0x00;
SYNCDELAY;
EP2FIFOCFG = 0x10;   // EP2 is AUTOOUT=1, AUTOIN=0, ZEROLEN=0, WORDWIDE=0
SYNCDELAY;
OUTPKTEND = 0x82;    // Arm both EP2 buffers to "prime the pump"
SYNCDELAY;
OUTPKTEND = 0x82;
… … … … …
```

Рис. 9-28. TD_Init Example: Configuring AUTOOUT = 1

```
TD_Init():
… … … … …
REVCTL = 0x03;       // MUST set REVCTL.0 and REVCTL.1 to 1
SYNCDELAY;
SYNCDELAY;
EP8CFG = 0xE0;       // EP8 is DIR=IN, TYPE=BULK
SYNCDELAY;
FIFORESET = 0x80;    // Reset the FIFO
SYNCDELAY;
FIFORESET = 0x08;
SYNCDELAY;
FIFORESET = 0x00;
SYNCDELAY;
EP8FIFOCFG = 0x0C;   // EP8 is AUTOOUT=0, AUTOIN=1, ZEROLEN=1, WORDWIDE=0
SYNCDELAY;
EP8AUTOINLENH = 0x02;  // Auto-commit 512-byte packets
SYNCDELAY;
EP8AUTOINLENL = 0x00;
… … … … …
```

Рис. 9-29. TD_Init Example: Configuring AUTOIN = 1

9.3.5 ДОСТУП CPU к OUT пакетам, AUTOOUT = 1

Когда **AUTOOUT = 1**, CPU чипа FX2 не находится на пути доступа к данным от хоста к мастеру. Для того чтобы достигнуть максимальной пропускной способности шины USB 2.0, хост и мастер непосредственно связаны друг с другом, обходя CPU.

Рис. 9-30 показывает, что в режиме **Auto-Out** данные из хоста автоматически принимаются в буферы FIFO без вмешательства микропрограммы микропроцессора.

```
TD_Poll():
… … … … …
// no code necessary to xfr data from host to master!
// AUTOOUT=1 and SIZE=0 auto-commits packets
// in 512 byte chunks.
… … … … …
```

Рис. 9-30. TD_Poll Example: Не нужен никакой код для пакетов OUT, когда AUTOOUT=1

9.3.6 Доступ CPU к OUT-пакетам, AUTOOUT = 0

В некоторых системах может быть желательно разрешить CPU FX2 участвовать в передаче данных между хостом и **Slave FIFO**. Для того чтобы сконфигурировать FIFO для этого режима «Manual-Out», в соответствующем регистре **EPxFIFOCFG** должен быть очищен (установлен в **'0'**) бит **AUTOOUT** (смотри **Рис. 9-31**).

```
TD_Init():
… … … … …
REVCTL = 0x03;       // MUST set REVCTL.0 and REVCTL.1 to 1
SYNCDELAY;
EP2CFG = 0xA2;       // EP2 is DIR=OUT, TYPE=BULK, SIZE=512, BUF=2x
SYNCDELAY;
FIFORESET = 0x80;    // Reset the FIFO
SYNCDELAY;
FIFORESET = 0x02;
SYNCDELAY;
FIFORESET = 0x00;
SYNCDELAY;
EP2FIFOCFG = 0x00;   // EP2 is AUTOOUT=0, AUTOIN=0, ZEROLEN=0, WORDWIDE=0
SYNCDELAY;
OUTPKTEND = 0x82;    // Arm both EP2 buffers to "prime the pump"
SYNCDELAY;
OUTPKTEND = 0x82;
… … … … …
```

Рис. 9-31. TD_Init Example, конфигурация AUTOOUT=0

Как показано на **Рис. 9-32**, микропрограмма FX2 может выполнить одну из трех операций, когда FX2 находится в режиме **«Manual-Out»** и пакет получен от хоста:

1. Она может передать (ввести в буферы FIFO) пакет, записывая **OUTPKTEND** с **SKIP=0** (Рис. 9-33).
2. Она может пропустить (отвергнуть) пакет, записывая **OUTPKTEND** с **SKIP=1** (Рис. 9-34).
3. Она может отредактировать пакет (или сгенерировать **[source]** целиком **OUT-**пакет), записывая в буфер FIFO непосредственно, затем записывая длину пакета в **EPxBCH:L**. Запись в **EPxBCL** вводит в действие отредактированный пакет, так что **EPxBCL** должен быть записан после записи **EPxBCH** (Рис. 9-35).

Во всех случаях, **OUT** буфер автоматически переактивизируется **[re-arm]**, так что он может получать следующий пакет.

Смотри Раздел 8.6.2.4 для подробного описания бита **SKIP**.

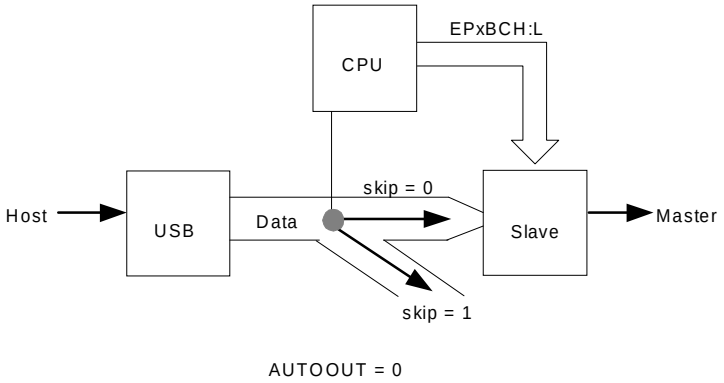

Рис. 9-32. Пропуск, передача или генерация пакета (AUTOOUT=0)

```
TD_Poll():
… … … … …
if( !( EP2468STAT & 0x01 ) )
{ // EP2EF=0 when FIFO NOT empty, host sent packet
  OUTPKTEND = 0x02;  // SKIP=0, pass buffer on to master
}
… … … … …
```

Рис. 9-33. TD_Poll Example, AUTOOUT=0, *ПЕРЕДАЧА* пакета

```
TD_Poll():
… … … … …
if( !( EP2468STAT & 0x01 ) )
{ // EP2EF=0 when FIFO NOT empty, host sent packet
  OUTPKTEND = 0x82;  // SKIP=1, do NOT pass buffer on to master
}
… … … … …
```

Рис. 9-34. TD_Poll Example, AUTOOUT=0, *ПРОПУСК* пакета

```
TD_Poll():
… … … … …
if( EP24FIFOFLGS & 0x02 )
{
SYNCDELAY;                //
FIFORESET = 0x80;         // nak all OUT pkts. from host
SYNCDELAY;                //
FIFORESET = 0x02;         // advance all EP2 buffers to cpu domain
SYNCDELAY;                //
EP2FIFOBUF[0] = 0xAA;     // create newly sourced pkt. data
SYNCDELAY;               //
EP2BCH = 0x00;
SYNCDELAY;                //
EP2BCL = 0x01;            // commit newly sourced pkt. to interface fifo

// beware of "left over" uncommitted buffers

SYNCDELAY;                //
OUTPKTEND = 0x82;         // skip uncommitted pkt. (second pkt.)
// note: core will not allow pkts. to get out of sequence
SYNCDELAY;                //
FIFORESET = 0x00;         // release "nak all"
}
… … … … …
```

Рис. 9-35. TD_Poll Example, AUTOOUT=0, *ГЕНЕРАЦИЯ НОВОГО* пакета

Если не переданный пакет находится в OUT-буфере конечной точки, когда FX2 сброшен, этот пакет не передается автоматически к мастеру. Для того чтобы проверить, что в буферах конечной точки нет никаких не переданных пакетов после сброса, подпрограмма «инициализации конечной точки» FX2 должна пропустить 2, 3 или 4 пакета (в зависимости от глубины буферизации, выбранной для FIFO), записывая ***OUTPKTEND*** *с* ***SKIP=1.*** *Смотри* ***Рис. 9-36.***

```
TD_Init():
… … … … …
REVCTL = 0x03;      // MUST set REVCTL.0 and REVCTL.1 to 1
SYNCDELAY;
SYNCDELAY;
EP2CFG = 0xA2;      // EP2 is DIR=OUT, TYPE=BULK, SIZE=512, BUF=2x
SYNCDELAY;
EP2FIFOCFG = 0x00;  // EP2 is AUTOOUT=0, AUTOIN=0, ZEROLEN=0, WORDWIDE=0

// OUT endpoints do NOT come up armed
SYNCDELAY;
OUTPKTEND = 0x82;   // arm first buffer by writing OUTPKTEND w/skip=1
SYNCDELAY;
OUTPKTEND = 0x82;  // arm second buffer by writing OUTPKTEND w/skip=1
… … … … …
```

Рис. 9-36. TD_Init Example, инициализация OUT Endpoint

9.3.7 Доступ CPU к IN-пакетам, AUTOIN = 1

РЕЖИМ **Auto-In** подобен режиму **Auto-Out:** когда **IN FIFO** сконфигурирован для режима **Auto-In** (устанавливая его бит **AUTOIN** в **'1'**), данные от мастера автоматически объединяются в пакеты и направляются к USB без какого-либо вмешательства CPU (смотри **Рис. 9-37**).

```
TD_Poll():
… … … … …
// no code necessary to xfr data from master to host!
// AUTOIN=1 and EP8AUTOINLEN=512 auto commits packets
// in 512 byte chunks.
… … … … …
```

Рис. 9-37. TD_Poll Example, AUTOIN = 1

Режим **Auto-In** отличается в одном важном аспекте от режима **Auto-Out:** в режиме **Auto-Out** данные (исключая данные в коротких пакетах) всегда автоматически передаются в 512- или 1024-байтовых пакетах; в режиме **Auto-In** размер автопередаваемого пакета может быть установлен любой величины, отличной от нуля (с единственным ограничением, конечно, что размер пакета должен быть меньше или равен размеру буфера конечной точки). Размер пакета **Auto-In** каждого буфера FIFO хранится в своей паре регистров **EPxAUTOINLEN[H:L].**

Чтобы получить **IN-**пакет, микропрограмма FX2 может временно остановить поток данных от внешнего мастера (обычно через сигнал на выводе входа/выхода общего назначения), дождаться, пока буфер конечной точки станет доступным, создать новый пакет, записывая непосредственно в этот буфер, затем направить пакет на USB и запустить внешний мастер. Таким образом, микропрограмма может включить свои собственные пакеты в поток данных.

Смотри Рис. 9-38, который иллюстрирует поток данных непосредственно между мастером и хостом, и Рис. 9-39, который показывает, как микропрограмма передает свой **IN-**пакет.

Пример микропрограммы показан на Рис. 9-40.

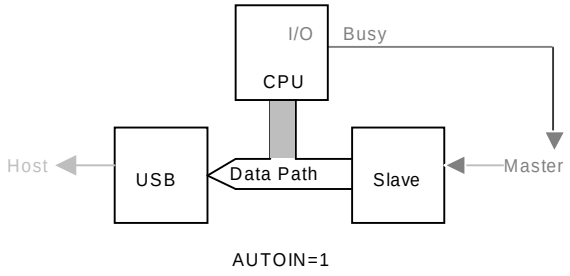

Рис. 9-38. Мастер пишет непосредственно в Хост, AUTOIN = 1

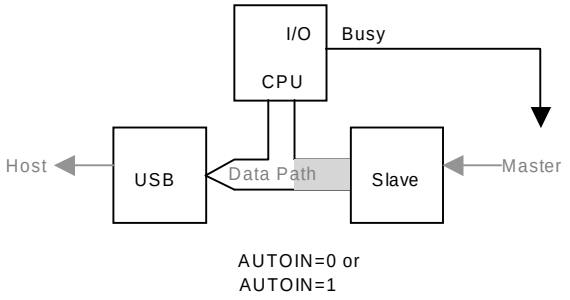

Рис. 9-39. Вмешательство микропрограммы, AUTOIN = 0 or 1

```
TD_Poll():
… … … … …
if( source_pkt_event )
{  // 100-msec background timer fired
  if( holdoff_master( ) )
  {  // signaled "busy" to master successful
    while( !( EP68FIFOFLGS & 0x20 ) )
    {  // EP8EF=0, when buffer not empty
      ;  // wait 'til host takes entire FIFO data
    }

    FIFORESET = 0x80;  // initiate the "source packet" sequence
    SYNCDELAY;
    FIFORESET = 0x06;
    SYNCDELAY;
    FIFORESET = 0x00;

    EP8FIFOBUF[ 0 ] = 0x02;  // <STX>, packet start of text msg
    EP8FIFOBUF[ 1 ] = 0x06;  // <ACK>
    EP8FIFOBUF[ 2 ] = 0x07;  // <HEARTBEAT>
    EP8FIFOBUF[ 3 ] = 0x03;  // <ETX>, packet end of text msg

    SYNCDELAY;
    EP8BCH = 0x00;
    SYNCDELAY;
    EP8BCL = 0x04;  // pass newly-sourced buffer on to host
  }
  else
  {
    history_record( EP8, BAD_MASTER );
  }
}
… … … … …
```

Рис. 9-40. TD_Poll Example: Передача IN-пакета

9.3.8 Доступ к IN-пакетам, AUTOIN=0

В некоторых системах, может быть желательным разрешать CPU FX2 участвовать в каждой передаче данных между внешним мастером и хостом. Для того чтобы сконфигурировать FIFO для этого режима **«Manual-In»,** бит **AUTOIN** в соответствующем регистре **EPxFIFOCFG** должен быть очищен (установлен вы **'0'**).

В режиме **Manual-In** микропрограмма FX2 может передавать, пропускать или редактировать пакеты, посылаемые внешним мастером, а также может непосредственно генерировать новые пакеты. Для того чтобы передать пакет, микропрограмма записывает номер конечной точки **(с SKIP=0)** в регистр **INPKTEND.** Для того чтобы пропустить пакет, микропрограмма записывает номер конечной точки с **SKIP=1** в регистр **INPKTEND.** Для того чтобы отредактировать или создать новый пакет, микропрограмма записывает в буфер FIFO, затем записывает длину пакета в **EPxBCH** и **EPxBCl** (в этом порядке).

```
TD_Poll():
… … … … …
if( master_finished_longxfr( ) )
{  // master currently points to EP8, pins FIFOADR[1:0]=11
  if( !( EP68FIFOFLGS & 0x10 ) )
  { // EP8FF=0 when buffer available
    INPKTEND = 0x08;  // firmware commits EP8 packet
                      // by writing 8 to INPKTEND
    release_master( EP8 );
  }
}
… … … … …
```

Рис. 9-41. TD_Poll Example, AUTOIN=0, ПЕРЕДАЧА пакета посредством INPKTEND

```
TD_Poll():
… … … … …
if( master_finished_longxfr( ) )
{  // master currently points to EP8, pins FIFOADR[1:0]=11
  if( !( EP68FIFOFLGS & 0x10 ) )
  { // EP8FF=0 when buffer available
    INPKTEND = 0x88;  // firmware skips EP8 packet
                      // by writing 0x88 to INPKTEND
    release_master( EP8 );
  }
}
… … … … …
```

Рис. 9-42. TD_Poll Example, AUTOIN=0, ПРОПУСК пакета посредством INPKTEND

```
TD_Poll():
… … … … …
if( master_finished_xfr( ) )
{  // modify the data
  EP8FIFOBUF[ 0 ] = 0x02;  // <STX>, packet start of text msg
  EP8FIFOBUF[ 7 ] = 0x03;  // <ETX>, packet end of text msg
  SYNCDELAY;
  EP8BCH = 0x00;
  SYNCDELAY;
  EP8BCL = 0x08;  // pass buffer on to host
}
… … … … …
```

Рис. 9-43. TD_Poll Example, AUTOIN=0, РЕДАКТИРОВАНИЕ пакета посредством EPxBCH:L

9.3.9 Инициализация Auto-In / Auto-Out

Разрешение Auto-In передачи между SLAVE FIFO и конечной точкой

Обычно FIFO конфигурируется для режима **Auto-In** следующим образом:

1. Сконфигурируйте биты **IFCONFIG[7:4]**, чтобы определить поведение тактов интерфейса.
2. Установите биты **IFCFG[1:0]=11.**
3. Сбросьте все буферы FIFO.
4. Установите бит **EPxFIFOCFG.3=1**.
5. Установите размер через регистры **EPxAUTOINLEN[H:L].**

Разрешение Auto-Out передачи между конечной точкой и SLAVE FIFO

Обычно FIFO конфигурируется для режима **Auto-Out** следующим образом:

1. Сконфигурируйте биты **IFCONFIG[7:4],** чтобы определять поведение тактов интерфейса.
2. Установите биты **IFCFG[1:0]=11.**
3. Сбросьте все буферы FIFO.
4. Установите бит **EPxFIFOCFG.4=1.**

9.3.10 Пример Auto-Mode: Синхронные FIFO IN передачи данных

```
TD_Init():
REVCTL = 0x03;   // MUST set REVCTL.0 and REVCTL.1 to 1
SYNCDELAY;
FIFORESET = 0x80;  // reset all FIFOs
SYNCDELAY;
FIFORESET = 0x02;
SYNCDELAY;
FIFORESET = 0x04;
SYNCDELAY;
FIFORESET = 0x06;
SYNCDELAY;
FIFORESET = 0x08;
SYNCDELAY;
FIFORESET = 0x00;
SYNCDELAY;          // this defines the external interface to be the following:
IFCONFIG = 0x43;   // use IFCLK pin driven by external logic (5MHz to 48MHz)
                  // use slave FIFO interface pins driven sync by external master
EP8FIFOCFG = 0x0C; // this lets the FX2 auto commit IN packets, gives the
                   // ability to send zero length packets,
                   // and sets the slave FIFO data interface to 8-bits
EP8CFG = 0xE0;     // sets EP8 valid for IN's
                   // and defines the endpoint for 512 byte packets, 2x buffered
PINFLAGSAB = 0x00; // defines FLAGA as prog-level flag, pointed to by FIFOADR[1:0]
SYNCDELAY;          // FLAGB as full flag, as pointed to by FIFOADR[1:0]
PINFLAGSCD = 0x00; // FLAGC as empty flag, as pointed to by FIFOADR[1:0]
                   // won't generally need FLAGD

PORTACFG = 0x00;   // used PA7/FLAGD as a port pin, not as a FIFO flag
FIFOPINPOLAR = 0x00; // set all slave FIFO interface pins as active low

SYNCDELAY;
EP8AUTOINLENH = 0x02; // you can define these as you wish,
SYNCDELAY;            // to have the FX2 automatically limit IN's
EP8AUTOINLENL = 0x00;

SYNCDELAY;
EP8FIFOPFH = 0x82; // you can define the programmable flag (FLAGA)
SYNCDELAY;          // to be active at the level you wish
EP8FIFOPFL = 0x00;

SYNCDELAY;          // out endpoints do not POR (power-on reset) armed
EP2BCL = 0x80;      // since the defaults are double buffered we must
SYNCDELAY;          // write dummy byte counts twice
EP2BCL = 0x80;      // arm EP2OUT & EP4OUT by writing to the byte count w/skip.
SYNCDELAY;
EP4BCL = 0x80;
SYNCDELAY;
EP4BCL = 0x80;
```

```
TD_Poll():
// nothing! The FX2 is doing all the work of transferring packets
// from the external master sync interface to the endpoint buffer...
```

Рис. 9-44. Пример Кода, Синхронные FIFO IN передачи данных (Synchronous Slave FIFO IN Data Transfer)

9.3.11 Пример Auto-Mode: Асинхронные FIFO IN передачи данных

Код инициализации точно такой же, как и для примера синхронной передачи в Разделе 9.3.10, но с **IFCLK**, сконфигурированным для внутреннего использования с частотой 48 МГЦ и битом **ASYNC**, установленным в **'1'**.

Рис. 9-45 показывает построчную модификацию, которая необходима.

```
TD_Init( ):  // slight modification from our synchronous firmware example
IFCONFIG = 0xCB;
// this defines the external interface as follows:
// use internal IFCLK (48MHz)
// use slave FIFO interface pins asynchronously to external master
```

Рис. 9-45. Пример TD_Init, Асинхронные Slave FIFO IN передачи данных

Код для выполнения передач данных, как и прежде, необязателен.
Это иллюстрирует Рис. 9-46.

```
TD_Poll( ):
// nothing!  The FX2 is doing all the work of transferring packets
// from the external master async interface to the endpoint buffer…
```

Рис. 9-46. Пример TD_Poll, Асинхронные Slave FIFO IN передачи данных

9.4 Переключения между Manual-Out и Auto-Out

Поскольку **Out** конечные точки автоматически не активизируются, когда FX2 вводит режим **Auto-Out,** микропрограмма может безопасно переключать FX2 между режимами **Manual-Out** и **Auto-Out** без всякой необходимости отключать или сбрасывать буферы FIFO.