Практикум программиста USB-устройств Часть 1. EZ-USB FX2LP – универсальное USB-решение

Дмитрий Чекунов (г. Ижевск)

Данный цикл статей является логическим продолжением цикла «Программисту USB-устройств». Теперь, когда есть теоретические знания о шине USB, надо применить их на практике. Но с чего начать? В какой последовательности действовать? Новый цикл статей даст ответы на подобные вопросы. Выполним мини-проект по разработке USB-устройства и последовательно пройдём все этапы, начиная от знакомства с элементной базой и заканчивая написанием собственного драйвера.

Вступление

Для сопряжения устройств с шиной USB существует множество микросхем, специализированных и универсальных, со встроенным микроконтроллером и без. И если подключение с использованием специализированных микросхем не вызовет затруднений, поскольку они обычно преобразуют поток данных к некоему классическому интерфейсу (USART, SPI и прочих), то использование «родной» USB-микросхемы может не только вызвать затруднения, но и сильно затормозить процесс разработки из-за слабой изученности данного интерфейса. Чем выгодно применение «родных» USB-микросхем? Во-первых, только они способны дать вам действительно высокую скорость передачи данных. Во-вторых, только с их помощью можно создать топологию USB, соответствующую идеологии вашего устройства. В конечном счёте, все мы осваивали интерфейс USART, шину ISA, и в те времена это тоже казалось сложно. Главное – начать...

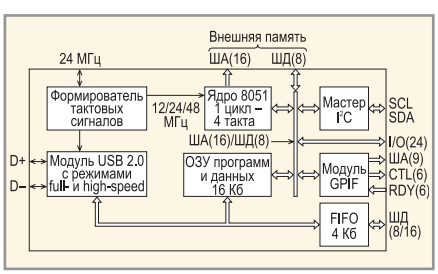


Рис. 1. Структура EZ-USB FX2LP

Итак, в качестве базовой микросхемы возьмём микроконтроллер CY7C68013A из семейства EZ-USB FX2LP фирмы Cypress [1]. Это новое семейство малопотребляющих микроконтроллеров с поддержкой интерфейса USB 2.0, его прародителем и почти полным аналогом является CY7C68013 из семейства EZ-USB FX2. Поэтому всё, что мы будем рассматривать далее, за исключением небольших деталей, в полной мере применимо к 68013.

Сразу отвечу на вопросы, интересующие программистов в первую очередь:

- *какая среда разработки используется для получения исполняемого кода?* Обычный ассемблер для микроконтроллеров 8051 с поддержкой включаемых файлов. Я использую ассемблер asem-51, доступный для свободного использования и скачивания через Интернет [2];
- *каким образом код «зашивается» в память программ?* Код загружается через USB, память программ представляет собой ОЗУ, поэтому заменять программу можно в любой момент, бесконечное количество раз – простор для экспериментов.

Обзор микроконтроллера

Микроконтроллер из семейства EZ-USB FX2LP (см. рис. 1) представляет собой высокоинтегрированную микросхему, построенную на базе быстродействующего ядра 8051, имеющую встроенную память объёмом 24 Кб и дополненную самыми разнообразными модулями, поддерживающими стандартные (USB, I²C) и универсальные интерфейсы (GPIF, FIFO) с внешними устройствами.

Микросхема питается напряжением 3,3 В и в любом рабочем режиме потребляет ток не более 85 мА, что позволяет использовать её в устройствах с питанием от шины USB.

Для работы микросхемы используется кварц с частотой 24 МГц. Формирователь тактовых сигналов, в соответствии с запрограммированным значением (перепрограммирование допускается в любой момент по ходу выполнения программы), выдаёт тактовый сигнал на ядро 8051 с частотой 12, 24 или 48 МГц. Для модуля USB формируется сигнал с частотой 480 МГц, что позволяет ему работать в высокоскоростном режиме.

Загрузка кода в FX2LP производится через интерфейс I²C или USB.

После прохождения сигнала Reset микроконтроллер проверяет наличие загрузочной микросхемы на шине I²C. В случае, если микросхема найдена, выполняется загрузка кода и подаётся внутренний сигнал Reset на ядро 8051. Устройство регистрируется на шине с новыми идентификаторами и дальше работает по заложенной программе.

В случае, если микросхема на шине I²C не найдена, ядро 8051 остается выключенным, а модуль USB регистрируется на шине как устройство с топологией, заданной по умолчанию (см. рис. 2). Теперь, после регистрации, возможна загрузка исполняемого кода через USB. Загрузка кода всегда заканчивается подачей внутреннего сигнала Reset на ядро 8051; далее, как правило, происходит отключение устройства от шины USB и повторное подключение уже с другими идентификаторами (VID, PID, DID). Дальней-

шая работа устройства определяется новой программой.

Обновление программы через USB доступно в любой момент. Память программ имеет тип ОЗУ, поэтому ограничений на число загрузок программного обеспечения нет.

Микроконтроллеры в корпусах с количеством выводов 100 и 128 имеют дополнительно 2 встроенных последовательных порта (USART0, USART1), дополнительные входы внешних прерываний (INT4, INT5, INT6), счётные входы для таймеров/счётчиков (T0, T1, T2), а также сигналы для управления внешним ОЗУ (RD, WR). Подключение внешней памяти программ возможно только для микроконтроллеров в 128-выводном корпусе.

Система прерываний

Для повышения эффективности работы система прерываний FX2LP значительно расширена. Помимо стандартных прерываний, унаследованных от 8051, дополнительными источниками для него являются: таймер T2, последовательный порт USART1, модули USB, I²C, GPIF/FIFO, внешние входы INT4, INT5, INT6 и событие RESUME. Перечень всех источников прерываний представлен в табл. 1.

Событие RESUME переводит микроконтроллер из спящего состояния в рабочее. Прерывание, генерируемое этим событием, имеет наивысший приоритет и является немаскируемым. Источниками, «пробуждающими» микроконтроллер, являются: внешние входы WAKEUP, WU2/PA3 и активность на шине USB. Индивидуальные разрешения и активные логические уровни для внешних входов устанавливаются в регистре WAKEUPCS.

Для всех остальных источников прерываний имеется программное изменение приоритетов и разрешение на обслуживание прерывания.

Особое внимание следует уделить прерываниям от модулей USB и GPIF/FIFO. Для каждого из этих прерываний характерно большое количество событий, формирующих запрос. В таблицах 2 и 3 в порядке убывания приоритета представлены события, контролируемые модулями USB и GPIF/FIFO соответственно. Как видим, их количество достаточно велико, поэтому для быстрого перехода на соответствующий вектор прерывания используется дополнительная таблица векторов прерываний совместно с системой автоматического перенаправления (autovectoring). Включение системы осуществляется битами AV2EN (для USB) и AV4EN (для GPIF/FIFO) в регистре INTSETUP. Тогда при возникновении прерывания происходит автоматическая идентификация его источника и загрузка программного счётчика соответствующим адресом из таблицы векторов, после чего выполняется переход на подпрограмму обслуживания.

Прерывание GPIF/FIFO/INT4 является разделяемым и используется входом INT4 или модулями GPIF/FIFO. Выбор источника прерывания осуществляется битом INT4SRC в регистре INTSETUP.

Организация памяти

Микроконтроллер FX2LP имеет стандартное для 8051 внутреннее ОЗУ объёмом 256 байт и SFR-регистры.

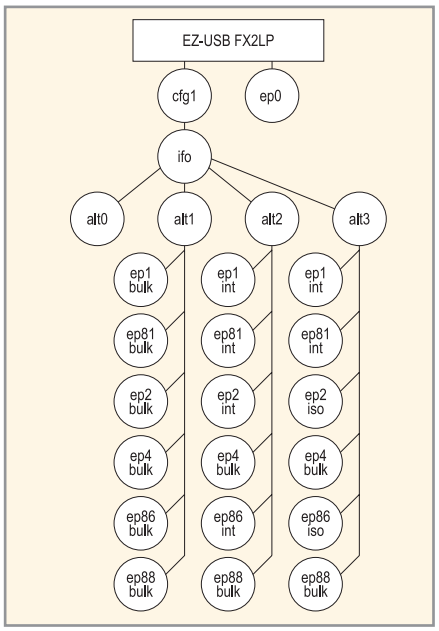


Рис. 2. Топология EZ-USB FX2LP по умолчанию

Таблица 1. Источники прерываний EZ-USB FX2LP

Приоритет	Источник прерывания	Возможность изменения приоритета	Вектор прерывания	Описание
0	RESUME	Нет	0033h	Пробуждение МК по событию RESUME
1	INT0	Да (IP.0)	0003h	Внешнее прерывание по входу INT0
2	T0	Да (IP.1)	000Bh	Прерывание от таймера T0
3	INT1	Да (IP.2)	0013h	Внешнее прерывание по входу INT1
4	T1	Да (IP.3)	001Bh	Прерывание от таймера T1
5	USART0	Да (IP.4)	0023h	Окончание приёма или передачи последовательным портом USAR
6	T2	Да (IP.5)	002Bh	Прерывание от таймера T2
7	USART1	Да (IP.6)	003Bh	Окончание приёма или передачи последовательным портом USART1
8	USB	Да (EIP.0)	0043h	Событие на шине USB
9	I²C	Да (EIP.1)	004Bh	Завершение операции на шине I²C
10	INT4/GPIF/FIFO	Да (EIP.2)	0053h	Внешнее прерывание по входу INT4 или прерывание от GPIF/FIFO
11	INT5	Да (EIP.3)	005Bh	Внешнее прерывание по входу INT5
12	INT6	Да (EIP.4)	0063h	Внешнее прерывание по входу INT6

Таблица 2. Источники прерываний GPIF/FIFO

Приоритет	Источник прерывания	Смещение в таблице векторов прерываний	Описание
1	EP2PF	80h	Заполненность FIFO точки 2 достигла запрограммированного значения
2	EP4PF	84h	Заполненность FIFO точки 4 достигла запрограммированного значения
3	EP6PF	88h	Заполненность FIFO точки 6 достигла запрограммированного значения
4	EP8PF	8Ch	Заполненность FIFO точки 8 достигла запрограммированного значения
5	EP2EF	90h	FIFO точки 2 пуст
6	EP4EF	94h	FIFO точки 4 пуст
7	EP6EF	98h	FIFO точки 6 пуст
8	EP8EF	9Ch	FIFO точки 8 пуст
9	EP2FF	0A0h	FIFO точки 2 заполнен
10	EP4FF	0A4h	FIFO точки 4 заполнен
11	EP6FF	0A8h	FIFO точки 6 заполнен
12	EP8FF	0ACh	FIFO точки 8 заполнен
13	GPIFDONE	0B0h	GPIF выполнил заданное количество транзакций
14	GPIFWF	0B4h	При выполнении транзакции установлен запрос на прерывание

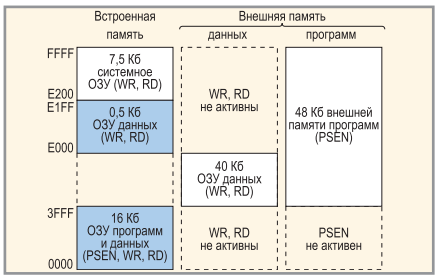


Рис. 3. Адресное пространство при EA = 0

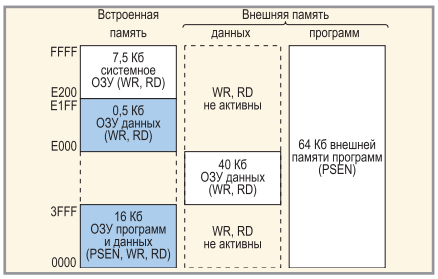


Рис. 4. Адресное пространство при EA = 1

Организация прочей памяти FX2LP сочетает в себе гарвардскую и фоннеймановскую архитектуру. В микроконтроллере имеется встроенное ОЗУ, воспринимаемое ядром как внешняя память данных (сегмент XDATA), в определённых случаях используемая как память программ.

Распределение и назначение встроенного ОЗУ зависит от количества выводов у микроконтроллера и схемы включения.

Так, адресное пространство для микроконтроллеров в 56- и 100-выводном корпусе ограничено только встроенным ОЗУ и представлено в левом столбце на рис. 3. Хотя микроконтроллер в 100-выводном корпусе и имеет сигналы WR и RD, отсутствие внешней шины данных лишает его возможности обращения к внешнему ОЗУ.

Распределение адресного пространства для микроконтроллера в 128-выводном корпусе, с входом EA, подключенным к корпусу, показано на рис. 3. При таком включении область встроенного ОЗУ (0000...03FFFh) используется как память данных и программ, а сигнал PSEN формируется только при обращении к адресам программы выше 03FFFh. Когда вход EA подключен к логической единице (рис. 4), вся встроенная память работает как память данных, а память программ располагается во внешней памяти.

Области встроенного ОЗУ, расположенные по адресам 0...3FFFh и 0E000h...0E1FFh (на рисунках выделены цветом), всегда доступны модулям USB и I²C для загрузки данных. Область 0E000h...0E1FFh всегда является памятью данных общего назначения. Системное ОЗУ, занимающее адреса 0E200h...0FFFFh, содержит регистры для управления встроенными модулями FX2LP, буферы контрольной и дополнительных точек, таблицу форм для GPIF.

Таблица 3. Источники прерываний модуля USB

Приоритет	Источник прерывания	Смещение в таблице векторов прерываний	Описание
1	SUDAV	00h	Данные, полученные в фазе SETUP, доступны в буфере
2	SOF	04h	Начало фрейма или микрофрейма
3	SUTOK	08h	Получен маркер запроса Setup
4	SUSPEND	0Ch	Получено требование на переход в режим пониженного потребления
5	USB RESET	10h	Получен сигнал Reset по шине USB
6	HISPEED	14h	Высокоскоростной режим работы доступен
7	EP0ACK	18h	FX2LP подтвердил фазу Control маркером ACK
8	Зарезервировано	1Ch	Не используется
9	EP0IN	20h	Буфер данных EP0 пуст
10	EP0OUT	24h	В буфер EP0 поступили данные
11	EP1IN	28h	Буфер EP1IN пуст
12	EP1OUT	2Ch	В буфер EP1OUT поступили данные
13	EP2	30h	IN: буфер пуст. OUT: в буфер поступили данные
14	EP4	34h	IN: буфер пуст. OUT: в буфер поступили данные
15	EP6	38h	IN: буфер пуст. OUT: в буфер поступили данные
16	EP8	3Ch	IN: буфер пуст. OUT: в буфер поступили данные
17	IBN	40h	На маркер запроса IN выдано подтверждение NAK
18	Зарезервировано	44h	Не используется
19	EP0PING	48h	На маркер запроса PING выдано подтверждение NAK
20	EP1PING	4Ch	На маркер запроса PING выдано подтверждение NAK
21	EP2PING	50h	На маркер запроса PING выдано подтверждение NAK
22	EP4PING	54h	На маркер запроса PING выдано подтверждение NAK
23	EP6PING	58h	На маркер запроса PING выдано подтверждение NAK
24	EP8PING	5Ch	На маркер запроса PING выдано подтверждение NAK
25	ERRLIMIT	60h	Количество ошибок на шине достигло ограничения
26	Зарезервировано	64h	Не используется
27	Зарезервировано	68h	Не используется
28	Зарезервировано	6Ch	Не используется
29	EP2ISOERR	70h	При изохронной передаче нарушена последовательность PID
30	EP4ISOERR	74h	При изохронной передаче нарушена последовательность PID
31	EP6ISOERR	78h	При изохронной передаче нарушена последовательность PID
32	EP8ISOERR	7Ch	При изохронной передаче нарушена последовательность PID

Организация и работа FIFO

Из встроенного ОЗУ FX2LP область, расположенная по адресам 0F000h...0FFFFh, организована как 8 блоков FIFO объёмом по 512 байт. Данное FIFO является буфером так называемых «больших» точек – EP2, EP4, EP6, EP8 – и используется для высокоскоростной передачи больших объёмов данных.

При конфигурировании точки программист выбирает размер FIFO (512 или 1024 байта) и глубину его буферизации (×2, ×3, ×4). Возможные варианты объединения блоков FIFO в домены и соответствующие им точки показаны на рис. 5. Несмотря на то что FIFO является буферизированным, обращение всегда происходит к адресам первого блока в домене. Например, передача данных через точку EP2IN с буфером, имеющим трёхкратную глубину, реально осуществляется так:

- записываем данные в первый блок и передаём его под управление модулю USB;
- повторяем запись в первый блок (реально записываем уже в следую-

щий) и передаём его под управление модулю USB;
- в третий раз заполняем первый блок (реально уже третий) и передаём его модулю USB.

После трёхкратного заполнения будет установлен признак полного буфера USB – флаг FULL в регистре EP2CS. Переключение буферов происходит аппаратно, и программист не знает номер реально используемого буфера в домене.

Для контроля состояния FIFO каждой точки используются следующие флаги: EPxEF – FIFO точки x пуст, EPxFF – FIFO точки x полон, EPxPF – FIFO точки x заполнен на запрограммированную величину. При установке соответствующего разрешения флаги генерируют запрос на прерывание. Все возможные источники прерываний для FIFO показаны в табл. 2.

FIFO имеет два режима работы: программный и аппаратный. В программном режиме ведущую роль играет сам микроконтроллер, который в соответствии с программой заполняет FIFO данными и передаёт его под управление модулю USB.

Второй режим подразумевает наличие некоторого аппаратного модуля, который самостоятельно заполняет FIFO и автоматически передаёт его модулю USB. В таком случае вмешательство микроконтроллера в работу FIFO не требуется, что позволяет достигать максимальной скорости передачи данных.

В FX2LP имеются два варианта аппаратного управления памятью FIFO. Первый вариант – это использование встроенного модуля GPIF (параллельный программируемый интерфейс), а второй вариант – подключение внешнего модуля управления памятью FIFO. Для подключения внешнего модуля имеются следующие сигнальные линии FIFO:
- тактовый сигнал – может быть как внешним, так и внутренним;
- шина данных – 8- или 16-разрядная;
- сигналы управления SLCS, SLWR, SLRD, SLOE, ADR0, ADR1 (номер точки), PKTEND;
- сигналы состояний FLAGA, FLAGB, FLAGC, FLAGD.

Модуль GPIF

GPIF – это мастер управления памятью FIFO с поддержкой программируемого параллельного интерфейса. Модуль GPIF предназначен для аппаратной передачи данных между буферами FIFO и внешними устройствами.

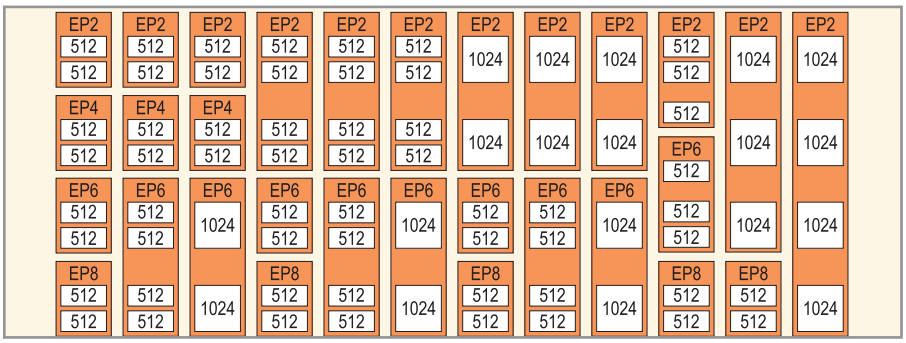


Рис. 5. Варианты объединения буферов FIFO в домены

Для подключения внешнего устройства модуль GPIF имеет следующие сигнальные линии:
- 9-разрядную шину адреса (охватывает адресное пространство объёмом 1024 байта – максимальный размер пакета USB);
- 6 выходных линий управления CTL0…CTL5;
- 6 входных контрольных линий RDY0…RDY5.

Последовательность формирования сигналов на линиях CTLx, контроль сигналов RDYx, изменение адреса, момент обмена данными – всё это задаётся программно и до начала работы заносится в таблицу форм. Перепрограммирование таблицы доступно в любое время.

Модуль GPIF поддерживает 4 режима работы:
- FIFO Read – пакетное чтение. GPIF самостоятельно заполняет FIFO данными и передает буфер для отправки. При заполнении всех буферов GPIF приостанавливает работу до появления свободного буфера. Вмешательство микроконтроллера не требуется;
- FIFO Write – пакетная запись. GPIF передаёт данные из FIFO внешнему устройству. При отсутствии данных в FIFO GPIF приостанавливает работу до прихода нового пакета. Вмешательство микроконтроллера не требуется;
- Single Read – одиночное чтение. Для активизации транзакции используется микроконтроллер. При обращении к регистрам XGPIFSGLDATH/L со стороны ЦПУ модуль GPIF формирует заданную последовательность сигналов, принимает данные от внешнего устройства и помещает их в названные регистры;
- Single Write – одиночная запись. Для активизации транзакции используется микроконтроллер. При записи данных в регистры XGPIFSGLDATH/L GPIF передаёт данные внешнему устройству, дополняя их последовательностью служебных сигналов.

За GPIF зарезервировано два прерывания (см. табл. 2): GPIFDONE и GPIFWF. Первое обычно используется при пакетном обмене данными и является признаком завершения передачи заданного объёма информации. GPIFWF – прерывание, устанавливаемое в процессе выполнения транзакции при совпадении определённых условий, используется в одиночных транзакциях.

Модуль GPIF позволяет реализовать такие интерфейсы, как ATA, UTOPIA и прочие.

Для улучшения характеристик модуля GPIF в FX2LP добавлен модуль коррекции ошибок ECC. Он позволяет контролировать данные, проходящие через GPIF, и корректировать ошибку в одном бите или обнаружить ошибку в двух битах блока данных.

Модуль USB-интерфейса

Теперь рассмотрим возможности встроенного модуля USB-интерфейса. Модуль поддерживает два режима работы шины USB из трёх возможных – полноскоростной (full-speed) и высокоскоростной (high-speed). Подключение модуля к шине USB происходит программно с помощью бита DISCON в регистре USBCS. Сразу после подключения происходит аппаратная идентификация доступного режима работы для FX2LP. В случае, когда хост предоставляет возможность работать в высокоскоростном режиме, модуль USB вырабатывает прерывание HISPEED, иначе модуль продолжает функционировать в полноскоростном режиме.

Как известно, далее хост запрашивает описание устройства. В FX2LP имеется встроенная поддержка стандартных требований. Программист управ-

ляет разрешением встроенной поддержки битом RENUM регистра USBCS [3]. В случае разрешения (RENUM = 0) микроконтроллер передаёт описание устройства, топология которого представлена на рис. 2. Если RENUM = 1, то необходимо самостоятельно обеспечить обслуживание стандартных требований, и в таком случае топология устройства полностью определяется программистом. Для упрощения передачи длинных дескрипторов имеется специальный указатель SUDPTR. При обслуживании требования GET_DESCRIPTOR в данный указатель достаточно загрузить адрес запрашиваемого описания, и вся последующая передача будет выполнена аппаратно.

Контрольная точка модуля имеет два буфера: один для пакета Setup, размером 8 байт, второй для дополнительных данных, передаваемых в одноименной фазе, размером 64 байта.

Кроме стандартных требований, модуль поддерживает дополнительные (vendor) требования. Фирма Cypress зарезервировала номера требований 0A0h…0AFh. Требование 0A0h используется для обмена данными со встроенным ОЗУ FX2LP, расположенным по адресам 0…3FFFh и 0E000h….0E1FFh. С помощью данного требования происходит загрузка нового кода для ядра 8051. Программист даже при отключении встроенной поддержки стандартных требований не может запретить выполнение зарезервированных требований, и поэтому для своих дополнительных требований необходимо использовать оставшиеся свободными номера.

Модуль USB поддерживает шесть точек. Точки EP1 и EP81 имеют буферы по 64 байта и считаются «малыми». Эти точки используются только для передач типа bulk и interrupt. Точки EP2, EP4, EP6, EP8 имеют в качестве буферов FIFO размером 512…1024 байта с двух-, трёх- или четырёхкратным буферизированием и считаются «большими» точками. Направление обмена «больших» точек программируется.

Источники прерывания модуля USB представлены в табл. 3.

Модуль I²C

FX2LP имеет встроенный модуль поддержки интерфейса I²C. Модуль работает только в режиме «мастер» на выбранной программно скорости – 100 или 400 КГц. Для работы используются всего три регистра: I2CS, I2DAT, I2CTL. Модуль генерирует прерывание после передачи или приёма байта по шине.

Помимо общего назначения – обмена данными с микросхемами I²C в процессе работы, данный модуль имеет ещё одно специфическое назначение. После прохождения внешнего сигнала Reset модуль используется для загрузки памяти программ из внешней микросхемы. Поиск загрузочной микросхемы происходит в следующем порядке:

- чтение данных из ячейки по адресу 0 микросхемы с адресом 0, для которой адрес ячейки задаётся одним байтом;
- чтение данных из ячейки по адресу 0 микросхемы с адресом 1, для которой адрес ячейки задаётся двумя байтами.

Если считанный байт имеет значение 0C0h, то происходит загрузка идентификационных значений (VID, PID, DID). Если считанный байт имеет значение 0C2h, то происходит загрузка программы. В конце любой загрузки происходит подача внутреннего сигнала Reset на ядро 8051.

Обзор программных инструментов

После знакомства с микроконтроллером перейдём к знакомству с программными инструментами, которые обеспечат взаимодействие с разрабатываемым USB-устройством и его «сердцем» – CY7C68013x. На начальном этапе нам потребуется поддержка выполнения следующих действий: трансляция программы в машинные коды, загрузка программного кода в микроконтроллер, передача USB-требований и обмен данными с заданной точкой устройства.

Выбор и настройка транслятора

Существует огромное количество трансляторов, которые позволяют получить код для микроконтроллеров, совместимых с 8051, поэтому предпочтение следует отдать хорошо изученному транслятору. Главное, чтобы он поддерживал включаемые файлы с предопределёнными именами регистров.

В частности, для решения нашей задачи подойдёт ассемблер asem-51 [2]. Его несомненными достоинствами являются свободное распространение и многоплатформенность.

Для того чтобы в будущем не заострять внимание на формате вызова транслятора, создадим простой пакетный файл, скрывающий особенности того или иного ассемблера. Основные требования к пакетному файлу:

- после завершения сообщить о результате выполнения операции;
- при успешном завершении создать файл с кодом программы для CY7C68013x;
- в случае возникновения ошибок создать файл с их полным перечнем.

Пример файла asm.bat:

```
@echo off
rem Задаём имя главного файла проекта
set NAME_PROJECT=mydevice
rem Вызываем ассемблер по заданному пути и с необходимыми параметрами
..\..\asem5113\asem %NAME_PROJECT%.asm >%NAME_PROJECT%.err
rem Анализируем результат завершения операции
if %ERRORLEVEL% geq 1 goto error
rem Сообщаем об успешном завершении
echo Трансляция завершена - Ok
Программа - %NAME_PROJECT%.hex
del %NAME_PROJECT%.err
goto end
:error
rem Сообщаем об ошибке
echo Трансляция завершена - ошибки
смотри в файле %NAME_PROJECT%.err
:end
rem Конец пакетного файла asm.bat
```

В третьей строке определено имя главного файла нашего низкоуровневого проекта – mydevice (например), это значит, что в текущем каталоге должен находиться главный файл проекта mydevice.asm.

В пятой строке пакетного файла происходит вызов ассемблера с указанием полного пути к нему; путь обязательно должен соответствовать реальному пути к ассемблеру на вашем компьютере.

В результате выполнения будет выдано сообщение об ошибке или об успешном завершении.

Готовый пакетный файл можно найти на сайте журнала.

Пакет программ для работы с CY7C68013x

Для связи с микроконтроллерами семейств EZ-USB FX2[LP] фирма

Cypress предоставляет разработчикам программный пакет – USB Developer's uStudio [4]. Пакет содержит универсальный, настраиваемый пользователем драйвер, библиотеки для работы с драйвером и программу управления USB-устройствами. Данный комплект программ позволяет разработчику практически сразу приступить к разработке и отладке низкоуровневого программного обеспечения.

Настройка драйвера

Драйвер является связующим звеном между приложением пользователя и USB-устройством. После инсталляции пакета USB Developer's uStudio установочные файлы драйвера располагаются в каталоге \Program Files\Cypress\USB DevStudio\Driver. К установочным файлам относятся:

- CyUSB.sys – собственно сам драйвер, совместимый с моделью WDM и предназначенный для работы под управлением операционной системы Windows 2000 или Windows XP;
- CyUSB.inf – информационный файл, определяющий рабочие параметры.

Изначально информационный файл не имеет сведений о каких-либо устройствах и действиях над ними, поэтому перед установкой драйвера необходимо отредактировать CyUSB.inf. Для того чтобы приступить к работе с микроконтроллером, нам достаточно включить его поддержку и предусмотреть поддержку разрабатываемого устройства. Итак, добавим в информационный файл следующие идентификаторы:

- VID=04B4h&PID=8613h – для поддержки микроконтроллера;
- VID=3112h&PID=1973h – для поддержки разрабатываемого устройства (например).

В любом текстовом редакторе открываем файл CyUSB.inf, находим раздел [Cypress], содержащий единственную строку следующего вида:

```
;%VID_XXXX&PID_XXXX.DeviceDesc%=CyUsb, USB\VID_XXXX&PID_XXXX,
```

и ниже добавляем две строки в аналогичном формате, но с указанием собственных идентификаторов вместо символов XXXX. В результате получим описание:

```
%VID_04B4&PID_8613.DeviceDesc%=CyUsb, USB\VID_04B4&PID_8613
%VID_3112&PID_1973.DeviceDesc%=CyUsb, USB\VID_3112&PID_1973
```

Сразу замечу, что строки, начинающиеся с символа точки с запятой (;), являются комментарием и в процессе работы игнорируются, поэтому в добавляемых строках названные символы необходимо удалить.

Далее, почти в самом конце файла, находим раздел [String] и в нём строку следующего содержания:

```
VID_XXXX&PID_XXXX.DeviceDesc="Cypress Generic USB Device".
```

Здесь происходит сопоставление идентификаторов устройства и названия. Добавляем ещё две строки следующего вида (XXXX заменяем реальными значениями):

```
VID_04B4&PID_8613.DeviceDesc="MCU CY7C68013x"
VID_3112&PID_1973.DeviceDesc="My First USB Device"
```

Заданные нами названия будут использованы системным диспетчером устройств и консольной программой CyConsole при отображении подключенных устройств.

Итак, установочные файлы драйвера готовы. Запомним каталог, в котором они располагаются, так как при первом подключении устройства необходимо будет произвести установку драйвера, указав полный путь к установочным файлам.

Возможности драйвера на этом не исчерпаны. Программисту доступны для использования в собственных программах функции CyAPI из библиотеки CyAPI.lib, обеспечивающие взаимодействие с драйвером CyUSB. Для более подробного знакомства с библиотечными функциями следует обратиться к руководству программиста Cypress CyAPI Programmer's Reference, входящего в состав пакета USB Developer's uStudio.

Файл CyUSB.inf, отредактированный в соответствии с рассмотренным примером, можно найти в на сайте журнала.

Обзор возможностей программы обслуживания

Теперь, когда драйвер подготовлен к установке, познакомимся с программой обслуживания USB-устройств. Программа CyConsole.exe входит в состав пакета USB Developer's uStudio и находится в каталоге \Program Files\Cypress\USB DevStudio\CyConsole. Программа имеет два режима работы:

1. Обмен данными с типовым USB-устройством;
2. Низкоуровневый доступ к функциям микроконтроллера CY7C68013x.

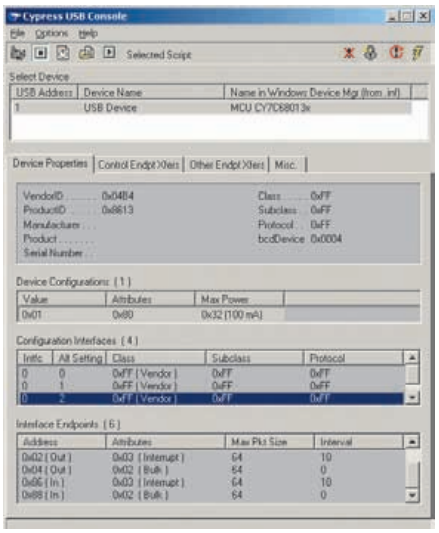


Рис. 6. CyConsole: закладка свойства устройства

Внешний вид консоли для первого режима работы представлен на рис. 6. Как можно заметить, интерфейс программы вполне дружелюбен, нагляден и интуитивно понятен. В таблице Select Device отображаются найденные устройства. По умолчанию программа работает с драйвером CyUSB, поэтому круг идентифицируемых устройств ограничен списком, составленным пользователем в информационном файле CyUSB.inf. На протяжении всего времени работы программы осуществляется контроль за подключением устройств, а любое изменение незамедлительно отражается в таблице Select Device.

Выбор устройства для работы осуществляется простым «кликанием мыши», при этом на закладке Device Properties будут показаны его свойства. Пользователю доступны для изменения следующие свойства: конфигурация, интерфейс и альтернативная установка. Изменение любого из параметров приводит к переходу устройства в новый режим работы, а точки, доступные в этом режиме, отображаются в таблице Interface Endpoints. Значения, установленные на первой закладке, становятся активными для работы на остальных закладках и не меняются до следующего выбора пользователя.

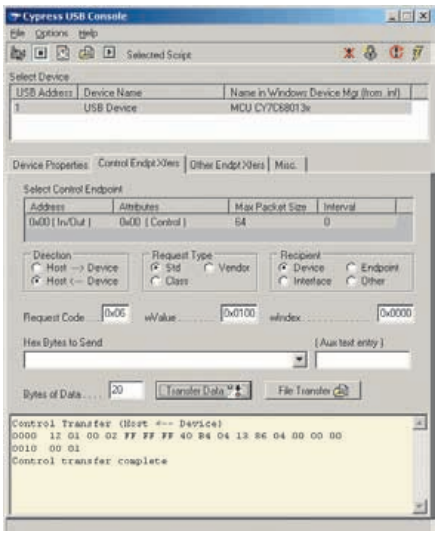


Рис. 7. CyConsole: закладка контрольных транзакций

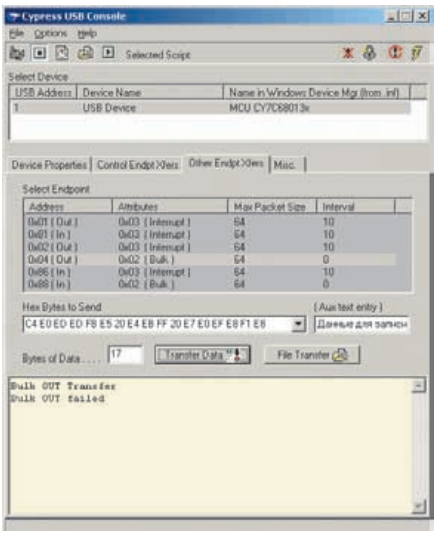


Рис. 8. CyConsole: закладка связи с точками USB-устройства

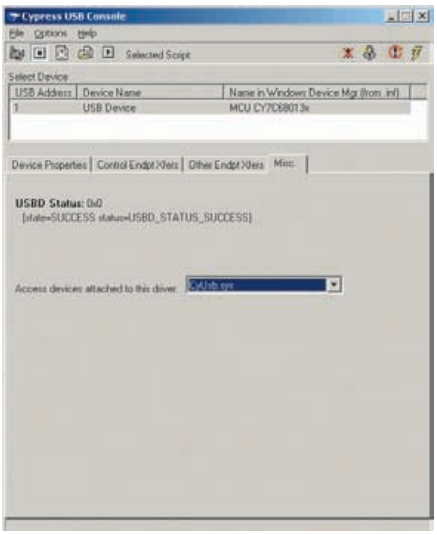


Рис. 9. CyConsole: закладка выбора активного драйвера

На рис. 7 представлена закладка Control Endpt Xfers, предназначенная для передачи USB-устройству как стандартных, так и дополнительных требований. Пользователь, используя элементы управления, имеет возможность сформировать 8-байтный пакет SETUP с любыми значениями полей. В примере, показанном на рис. 7, выполнено стандартное требование GET_DESCRIPTOR, с помощью которого получено описание устройства.

Закладка Other Endpt Xfers, показанная на рис. 8, предназначена для обмена данными с некоторой точкой. Набор доступных точек соответствует альтернативной установке, выбранной на первой закладке. Направление передачи данных определяется адресом точки.

Для передачи данных точке с направлением OUT пользователь должен ввести данные в шестнадцатеричном виде в строку Hex Bytes to Send или в текстовом виде в строку Aux text entry и нажать кнопку <Transfer Data>. Для передачи данных из файла используется кнопка <File Transfer>, после её нажатия появится диалоговое окно выбора файла данных.

Перед обменом данными с точкой IN необходимо задать количество принимаемых байт в строке Bytes of Data и нажать кнопку <Transfer Data>, принятые данные будут показаны в информационном окне. Для приёма и сохранения данных в файл необходимо использовать кнопку <File Transfer>.

В примере на рис. 8 передача данных точке с адресом 4 завершилась с ошибкой. Буфер точки занят (Bulk out failed), поэтому данные не достигли адресата.

Последняя закладка, показанная на рис. 9, может понадобиться в случае, если программист использует расширенные возможности драйвера CyUSB и среди нескольких драйверов требуется выбрать определённый.

Второй режим работы программы предоставляет низкоуровневый доступ к функциям микроконтроллера. Для перехода в этот режим необходимо выбрать пункты меню Options → → EZ–USB Interface. На экране появится окно, представленное на рис. 10. Среди огромного количества кнопок управления выделим наиболее значимые:

- <Clear> – очистить окно с информацией;
- <Download> – загрузить программу в ОЗУ микроконтроллера;
- <Lg EEPROM> – записать закодированную программу в загрузочную микросхему EEPROM с двухбайтным адресом;
- <S EEPROM> – записать закодированные данные/программу в загрузочную микросхему EEPROM с однобайтным адресом.

Остальные кнопки предназначены для передачи различных требований контрольной точке и обмена данными с дополнительными точками, то есть дублируют закладки, представленные на рис. 7 и 8. Все результаты выполнения команд сохраняются в информационном окне, как показано на рис. 10, где отображены сведения об устройстве, полученные при нажатии кнопки <Get Dev>.

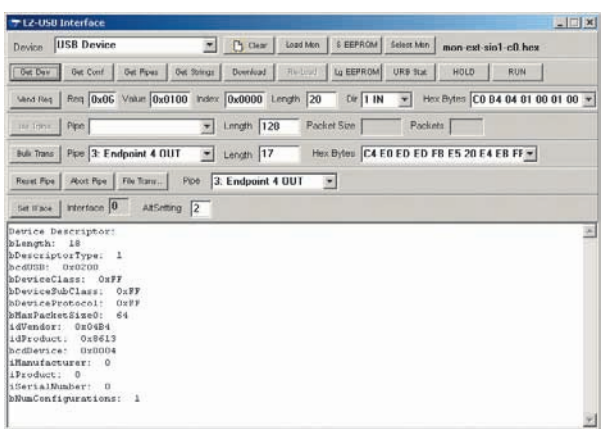


Рис. 10. CyConsole: режим доступа к функциям микроконтроллера

Литература

1. CY7C68013A/CY7C68015A EZ-USB FX2LP USB Microcontroller High-Speed USB Peripheral Controller. www.cypress.com.
2. http://plit.de/asem-51/final13.htm.
3. EZ-USB FX2 Technical Reference Manual. www.cypress.com.
4. ftp://ftp.efo.ru/pub/cypress/usb/USBDevStudio_1031.exe.

Уважаемый читатель!

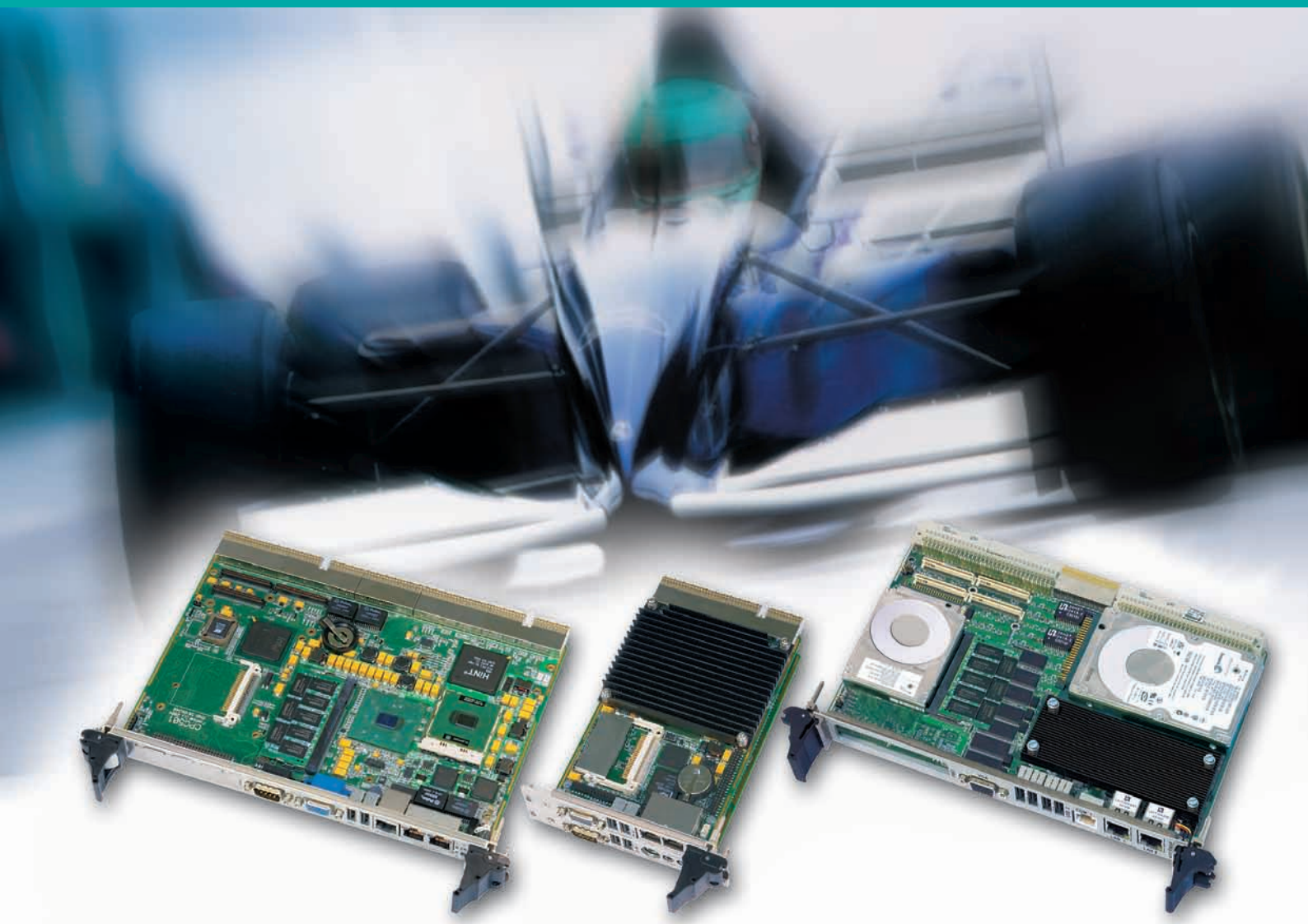
Ваши пожелания и замечания предлагаю оставлять на форуме сайта журнала «Современная электроника». Статьи цикла готовятся на «ходу», поэтому при возникновении у читателей большого количества вопросов темы могут быть скорректированы в соответствии с ситуацией. Думаю, что живой диалог с читателем увеличит качество и ценность предоставляемого материала.

С уважением,

Дмитрий Чекунов

Fastwel

На новом витке технологий!



Процессорные платы CompactPCI и VME
с процессором Pentium M

CPC501
Для телекоммуникаций
- Формат CPCI, 6U, 4HP
- Процессор Intel Pentium M 1,6 ГГц
- ОЗУ до 1 Гбайт DDR ECC
- Видеосистема с разрешением QXGA
- 2×Gigabit Ethernet, 1×Fast Ethernet
- 5×USB, 4×COM
- Слот PMC

CPC502
Для контрольно-измерительных систем
- Формат CPCI, 3U, 4/8/12HP
- Процессор Intel Pentium M 1,6 ГГц
- ОЗУ 512 Мбайт DDR ECC
- Видеосистема с разрешением QXGA
- 2×Gigabit Ethernet
- 2×Serial ATA
- 4×USB, 4×COM
- Поддержка PXI 2.1

CPC600
Для специальных систем управления
- Формат VME 64X, 6U
- Процессор Pentium M до 2,0 ГГц
- ОЗУ до 2 Гбайт DDR ECC
- Видеосистема с разрешением QXGA
- 4×Gigabit Ethernet
- 2×SerialATA
- 4×USB 2.0
- Слот PMC 64 бит

Диапазон рабочих температур: –40...+85°C (0...+70°C по запросу)

Удар: до 15g **Вибрация: до 2g**

ProSoft®

Официальный дистрибьютор Fastwel в России, странах СНГ и Балтии — компания ПРОСОФТ

МОСКВА Телефон: (095) 234-0636 • Факс: (095) 234-0640 • E-mail: info@prosoft.ru • Web: www.prosoft.ru

С.-ПЕТЕРБУРГ Телефон: (812) 325-3790 • Факс: (812) 325-3791 • E-mail: root@spb.prosoft.ru • Web: www.prosoft.ru

ЕКАТЕРИНБУРГ Телефон/факс: (343) 376-2820/376-2830 • E-mail: info@prosoftsystems.ru • Web: www.prosoftsystems.ru