Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi. Часть 2

Дмитрий ИОФФЕ
support@actel.ru
Андрей МАКСИМОВ
maksimov@actel.ru

Это вторая статья из цикла, посвященного применению микропроцессорного IP-ядра 8051s для ПЛИС фирмы Microsemi. В первой части статьи [1] было рассмотрено построение аппаратной части системы на основе 8051s с использованием IP-ядер, поставляемых в составе САПР Libero. Во второй части описывается микроконтроллерное ядро 8051s с точки зрения программиста.

IP-ядро 8051s от Microsemi для программиста

Прежде чем продолжить работу над проектом, нам нужно ознакомиться с тем, как выглядит ядро 8051s с точки зрения программиста. Этот раздел представляет собой краткий пересказ части текста [2].

Микроконтроллер 8051s очень похож на классический микроконтроллер семейства MCS-51, которое стало промышленным стандартом. Он имеет гарвардскую архитектуру, то есть раздельные адресные пространства для программы и данных. Память микроконтроллера делится на три области (рисунок):

- память программ (внутреннее ОЗУ, внешнее ОЗУ или внешнее ПЗУ);
- внешняя память данных (внешнее ОЗУ);
- внутренняя память данных (внутреннее ОЗУ).

Слова «внешняя» и «внутренняя» следует понимать применительно к микроконтроллерному ядру (то есть память находится внутри или снаружи ядра), а не к ПЛИС, внутри которой реализована микроконтроллерная система. «Внешнюю» память любого вида можно реализовать как внутри ПЛИС, так и вне ее. Во втором случае нужно будет вывести интерфейс внешней памяти ExternalMemIF [1] на выводы ПЛИС.

Доступ к внешней памяти

В интерфейсе внешней памяти ExternalMemIF линии адреса и данных для памяти программ и внешней памяти данных одни и те же. Если активен сигнал MEMPSRD, то в данный момент происходит обращение к памяти программ. Ядро 8051s может адресовать 64 кбайт памяти программ, от 0000h до FFFFh. Чтение из памяти программ происходит тогда, когда процессор выбирает очередной код операции или же когда выполняется команда ***MOVC***. После сброса процессор начинает выполнять программу с команды, расположенной по адресу 0000h. Нижняя часть памяти программ содержит векторы прерываний и сброса. Векторы прерываний расположены с интервалом в 8 байт, начиная с адреса 0003h. Память программ может быть реализована как внутреннее ОЗУ, внешнее ОЗУ, внешнее ПЗУ или как комбинация этих вариантов. Запись во внешнюю память программ поддерживается только в отладочном режиме, с использованием встроенного отладочного модуля (On-Chip Instrumentation block, OCI) и внешней отладочной аппаратуры с программным обеспечением.

Если же в интерфейсе внешней памяти активен один из сигналов — MEMWR или MEMRD, то это значит, что в данный момент происходит обращение к внешней памяти данных. Диапазон адресов внешней памяти данных для 8051s также составляет до 64 кбайт, от 0000h до FFFFh. Из них верхние 4 кбайт, от F000h до FFFFh, занимает шина APB, а нижние 60 кбайт отображаются на шину интерфейса внешней памяти. Запись во внешнюю память данных происходит при выполнении команд ***MOVX @Ri,A*** или ***MOVX @DPTR,A***, а чтение из нее — при выполнении команд ***MOVX A,@Ri*** или ***MOVX A,@DPTR***.

Время обращения к памяти программ или внешней памяти данных зависит от конфигурации параметров интерфейса внешней памяти 8051s [1] Program Memory Access и External Data Memory Access соответственно.

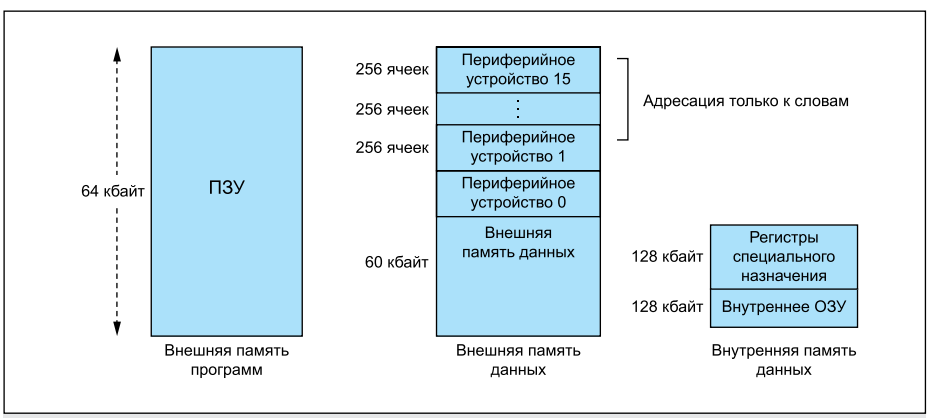

Рисунок. Организация памяти 8051s

Доступ к шине APB

Ядро 8051s соединяется с периферийными устройствами через шину APB, при этом оно является ведущим устройством на этой шине. Шина может иметь 8, 16 или 32 разряда данных, в соответствии с разрядностью подключенных к ней периферийных устройств. 8051s — 8-разрядный процессор,

поэтому для обращения к 16- или 32-разрядной шине в нем предусмотрены новые регистры специальных функций (SFR), так называемые X-регистры. Например, для записи данных в 32-разрядное устройство на шине APB программа, работающая на 8051s, должна сначала выполнить три отдельных 8-разрядных цикла записи в X-регистры XWB1, XWB2 и XWB3. Когда программа выполняет запись байта по требуемому адресу на шине APB, этот байт будет выставлен на линиях PWDATA [7:0], а данные из X-регистров — на линиях PWDATA[31:8].

16- и 32-разрядные операции чтения данных с шины APB происходят аналогично. Чтобы прочитать 32-разрядное слово из ячейки APB, программа должна обратиться по адресу этой ячейки и получить непосредственно младший байт PRDATA[7..0]. Затем программа должна прочитать три X-регистра (XRB1, XRB2 и XRB3), чтобы получить оставшиеся биты данных [31:8], которые были защелкнуты в этих регистрах во время обращения к шине APB.

Для 4 кбайт адресного пространства, занимаемых шиной APB, возможен только пословный доступ. Размер слова может составлять 8, 16 или 32 бита в зависимости от реализации шины.

Обычно интерфейс APB ядра 8051s подключается к ядру CoreAPB3. К нему можно подключить до 16 периферийных устройств, таких как CoreTimer или CoreGPIO. Часто программно-доступные регистры этих периферийных устройств располагаются в адресном пространстве с интервалом в 4 байт. Это означает, что соседние регистры лежат по адресам со смещениями 0x00, 0x04, 0x08, 0x0C и т. д. Программа для Core8051s должна учитывать это. Например, для последовательного доступа к ячейкам периферийного устройства, подключенного к слоту 0 ядра CoreAPB3, программа должна обращаться по адресам 0xF000, 0xF004, 0xF008, 0xF00C и т. д.

Если периферийные устройства спроектированы описанным образом, то в адресном пространстве шины APB можно использовать только каждый четвертый адрес. То есть если все 4 кбайт адресного пространства APB заняты такими устройствами, то можно использовать только 1024 независимо адресуемые ячейки или, если подразумевается CoreAPB3, 64 ячейки на устройство.

Заметим, что разрядность данных шины APB не зависит от схемы адресации. Каждая ячейка может хранить 8-, 16- или 32-разрядные данные. Разрядность данных APB задается во время конфигурирования ядра 8051s и должна быть не меньше максимальной разрядности данных подключаемых устройств.

Внутренняя память данных

Внутреннее ОЗУ

Адресное пространство внутренней памяти обслуживает 256 байт ОЗУ и 128 байт регистров специальных функций. Адрес внутренней памяти всегда восьмиразрядный, то есть размер адресного пространства составляет 256 байт (от 00h до FFh). Доступ к нижним 128 байт внутреннего ОЗУ возможен при помощи как прямой (direct), так и косвенной (indirect) адресации, а к верхним 128 байт — только через косвенную адресацию.

Нижние 128 байт содержат рабочие регистры и побитно адресуемую память. Нижние 32 байт образуют четыре банка по восемь регистров (R0–R7). Два бита слова состояния памяти программ (Program memory Status Word, PSW) определяют, какой из банков используется. Следующие 16 байт образуют блок побитно адресуемой памяти с адресами битов от 00h до 7Fh.

Регистры специальных функций

Регистры специальных функций (SFR) занимают верхние 128 байт внутренней памяти данных. Они доступны только через прямую адресацию. Таблица 1 содержит список SFR, представленных в Core8051s.

Таблица 1. Регистры специальных функций 8051s

Регистр	Адрес	Описание
SP	0x81	Указатель стека (Stack pointer)
DPL	0x82	Младший байт указателя данных (Data pointer) 0
DPH	0x83	Старший байт указателя данных 0
DPL1	0x84	Младший байт указателя данных 1 (опционально)
DPH1	0x85	Старший байт указателя данных 1 (опционально)
ICON	0x88	Регистр управления прерываниями
DPS	0x92	Выбор указателя данных (опционально)
XWB1	0x9A	Внешний буфер записи 1 (опционально)
XWB2	0x9B	Внешний буфер записи 2 (опционально)
XWB3	0x9C	Внешний буфер записи 3 (опционально)
XRB1	0x9D	Внешний буфер чтения 1 (опционально)
XRB2	0x9E	Внешний буфер чтения 2 (опционально)
XRB3	0x9F	Внешний буфер чтения 3 (опционально)
IE	0xA8	Регистр разрешения прерываний
PSW	0xD0	Слово состояния программы (адресуется побитно)
ACC	0xE0	Аккумулятор (адресуется побитно)
B	0xF0	Регистр B (адресуется побитно)

В таблице 1 приведено минимальное подмножество регистров специальных функций (SP, DPL, DPH, PSW, ACC и B), которое требуется для поддержки существующих компиляторов языка C. Здесь же показан опциональный второй указатель данных, недоступный по умолчанию. Кроме того, мы видим здесь шесть нестандартных SFR — описанных выше X-регистров. Регистры XWB1 и XRB1 представлены только тогда, когда константа APB_DWIDTH равна 16 или более. XWB2, XWB3, XRB2 и XRB3 представлены только при APB_DWIDTH, равном 32. Эти регистры используются при выполнении команды ***MOVX*** для доступа к адресному пространству шины APB. Эти шесть регистров не поддерживают побитовую адресацию.

Отметим также, что X-регистры доступны как для записи, так и для чтения. Это необходимо в ситуации, когда подпрограмма обработки прерывания требует доступа к шине APB, но прерывание произошло между обращением из основной программы к X-регистрам и выполнением команды ***MOVX***. В подобных случаях подпрограмма обработки прерывания должна в начале выполнения прочитать содержимое X-регистров и сохранить его, а по окончании — восстановить их исходные значения.

Аккумулятор (ACC)

Регистр ACC — это аккумулятор. Большинство команд использует аккумулятор для хранения операндов. В мнемониках для команд, обращающихся к аккумулятору, он обозначается как A, а не ACC.

Регистр B

Регистр B применяется в операциях умножения и деления. Он может также использоваться для временного хранения данных.

Слово состояния программы (PSW)

Таблица 2 содержит список битов слова состояния программы и их функции. В таблице 3 приведены параметры для выбора банка регистров при помощи флагов.

Таблица 2. Биты слова состояния программы (PSW)

Бит	Символ	Назначение
7	cy	Флаг переноса
6	ac	Флаг дополнительного переноса для двоично-десятичной арифметики
5	f0	Флаг общего назначения, доступный пользователю
4	rs1	Старший бит выбора банка рабочих регистров
3	rs0	Младший бит выбора банка рабочих регистров
2	ov	Флаг переполнения
1	-	Флаг, определяемый пользователем
0	p	Флаг четности, устанавливается аппаратно и указывает на четное число единиц в аккумуляторе

Таблица 3. Выбор банка регистров при помощи флагов rs1 и rs0

rs1, rs0	Выбранный банк	Адреса
00	Банк 0	00h — 07h
01	Банк 1	08h — 0Fh
02	Банк 2	10h — 17h
03	Банк 3	18h — 1Fh

Указатель стека (SP)

Указатель стека — это однобайтовый регистр. После сброса в него записывается адрес 07h. Этот регистр инкрементируется перед выполнением команд ***PUSH*** и ***CALL***.

Указатель данных (DPTR)

Указатель данных — двухбайтовый регистр. Его младший байт называется DPL, а старший — DPH. Его можно загружать как двухбайтовый регистр (***MOV DPTR,#data16***) или как два отдельных регистра (например, ***MOV DPL,#data8***). Обычно он используется для доступа к внешней памяти программ или данных (например, ***MOVC A,@A+DPTR*** или ***MOV A,@DPTR*** соответственно).

Программный счетчик (PC)

Программный счетчик располагается в двух байтах. После сброса в нем содержится

0000h. Он инкрементируется во время выборки из памяти программ кода команды или данных для команды.

Регистр разрешения прерываний (IE)

Регистр разрешения прерываний — однобайтовый. После сброса он содержит значение 00h. В таблице 4 приведено описание его битов. Обратите внимание на то, что оба бита — EAL и EX0 — должны быть установлены в «1», чтобы разрешить прерывание INT1.

Таблица 4. Назначение битов регистра разрешения прерываний

Бит	Символ	По умолчанию	Назначение
7	EAL	0	0 — запрет всех прерываний
6	–	0	Не используется
5	–	0	Не используется
4	–	0	Не используется
3	–	0	Не используется
2	EX1	0	0 — запрет внешнего прерывания 1 (INT1)
1	–	0	Не используется
0	EX0	0	0 — запрет внешнего прерывания 0 (INT0)

Регистр управления прерываниями (ICON)

Это однобайтовый регистр, после сброса в нем 00h. В таблице 5 показано назначение его битов. Регистр ICON представляет собой подмножество регистра управления таймером TCON, который обычно присутствует в реализациях процессора 8051.

Таблица 5. Регистр управления прерываниями

Бит	Символ	По умолчанию	Назначение
7	–	0	Не используется
6	–	0	Не используется
5	–	0	Не используется
4	–	0	Не используется
3	IE1	0	Флаг прерывания 1. Когда IT1=0, этот флаг отслеживает уровень на входе INT1. Когда IT1=1, этот флаг устанавливается по нарастающему фронту на входе INT1 и сбрасывается после обработки прерывания
2	IT1	0	Бит управления типом прерывания 1. Этот бит определяет, какое событие на входе INT1 вызывает прерывание: нарастающий фронт или высокий уровень. 0 — высокий уровень вызывает прерывание. 1 — нарастающий фронт вызывает прерывание
1	IE0	0	Флаг прерывания 0. Когда IT=0, этот флаг отслеживает уровень сигнала на входе INT0. Когда IT=1, этот флаг устанавливается по нарастающему фронту на входе INT0 и сбрасывается после обработки прерывания
0	–	–	Бит управления типом прерывания 0. Этот бит определяет, какое событие на входе INT0 вызывает прерывание: нарастающий фронт или высокий уровень. 0 — высокий уровень вызывает прерывание. 1 — нарастающий фронт вызывает прерывание

Прерывания

У ядра Core8051s есть два входа прерываний: INT0 и INT1. INT0 имеет низкий приоритет (уровень приоритета 0), с вектором по адресу 03h. INT1 имеет высокий приоритет (уровень приоритета 1), адрес вектора — 13h.

Регистры разрешения прерываний IE и управления прерываниями ICON задают поведение процессора при возникновении прерываний. В таблицах 4 и 5 показаны возможные варианты управления.

Блок OCI

Блок встроенного оборудования (On-Chip Instrumentation, OCI) подключается к внешней отладочной аппаратуре и ПО и помогает пользователю отлаживать программу. Блок OCI может быть включен опционально [1].

В ядре 8051s предусмотрены следующие возможности для отладки:

- управление запуском и остановкой;
- пошаговый режим;
- программное прерывание;
- выполнение программы отладчика;
- аппаратная точка прерывания;
- трассировка программы;
- доступ к аккумулятору.

Набор команд

Команды ядра 8051s совместимы на уровне двоичного кода с командами стандартного 8051 и выполняют те же функции. Это так называемый набор команд ASM51. Однако некоторые из этих команд недоступны по умолчанию, и при необходимости их следует специально подключать во время конфигурирования ядра [1]. Мы не будем здесь рассматривать набор команд 8051s, так как он подробно описывается в [3].

Поддержка компиляторов языка C

Так как ядро 8051s полностью совместимо с набором команд ASM51 и поддерживает три традиционных адресных пространства микроконтроллера 8051, оно может быть целевым процессором существующих компиляторов языка C для 8051. В этом разделе мы рассмотрим подробности, связанные с написанием кода на языке C при использовании компилятора Keil Cx51 или «Компилятора C для малых устройств» (Small Device C Compiler, SDCC), который поставляется в комплекте с ПО SoftConsole корпорации Microsemi.

Совместимость с ANSI C

Теоретически возможно написать для ядра 8051s код, полностью совместимый со стандартом ANSI C. Однако этому мешают несколько проблем:

- Некоторые типы аргументов функций в библиотеке времени выполнения выбранного нами компилятора изменены по сравнению с определениями стандарта ANSI C. Это сделано для уменьшения размера кода там, где это возможно.
- Некоторые функции в библиотеке времени выполнения нашего компилятора используют проприетарные расширения C, такие как типы bit и xdata.
- Некоторые функции, определенные в ANSI C, не представлены в библиотеке времени выполнения нашего компилятора.
- Библиотека времени выполнения нашего компилятора содержит некоторые внешние функции, не определенные в ANSI C.

Таким образом, код на «чистом» ANSI C будет гарантированно выполняться, если в нем при использовании библиотеки времени выполнения нашего компилятора не будут применяться упомянутые функции. В качестве альтернативы можно использовать другую библиотеку времени выполнения.

Однако большинство пользователей при необходимости просто переписывает свой код на ANSI C так, чтобы оптимально использовать архитектуру 8051.

Размещение переменных в C

В каком из трех адресных пространств системы 8051 размещаются переменные? По умолчанию, если не используются расширения C, все переменные размещаются в одном из адресных пространств, поэтому никакой путаницы не возникает. Обычно компилятор позволяет пользователю выбрать одну из трех возможных моделей памяти. Это малая (small), компактная (compact) и большая (large) модели. При работе с ядром 8051s наиболее интересны малая и большая модели.

Малая модель

В этой модели все переменные по умолчанию размещаются во внутренней памяти данных. При этом доступ к переменным получается наиболее быстрым. Однако все объекты (если их явным образом не расположить в другой области памяти), а также стек должны уместиться во внутреннем ОЗУ. Наиболее критичен здесь размер стека, так как от него зависит глубина вложения функций.

Большая модель

В большой модели по умолчанию все переменные располагаются во внешней памяти данных, размер которой может достигать 64 кбайт. Для ядра 8051s это 60 кбайт внешнего ОЗУ и 4 кбайт отведенной на память периферии. Для адресации внешней памяти используется указатель данных (Data Pointer, DPTR), поэтому доступ к переменным происходит медленнее, чем в малой модели. Вполне вероятно, однако, что для работы с 8051s подойдет именно большая модель, так как она обеспечивает доступ к периферийным устройствам без использования расширений языка C.

Проприетарные расширения языка C для 8051

Как мы уже сказали, существует возможность написать программу на переносимом ANSI C. Однако большинство пользователей используют нестандартные расширения, предлагаемые в различных компиляторах C, чтобы лучше адаптироваться к особенностям

архитектуры 8051. В частности, это касается разрядности адресов и данных, а также использования разных областей памяти.

Типы памяти

В таблице 6 показаны для примера некоторые типы памяти, которые используются в расширениях C для 8051.

Таблица 6. Типы памяти, используемые в расширениях C для 8051

Тип памяти	Описание
code	Память программ (64 кбайт). Доступна через команды MOVC @A + DPTR
data	Непосредственно адресуемая внутренняя память данных (128 байт). Она обеспечивает наиболее быстрый доступ
idata	Косвенно адресуемая внутренняя память данных. Для доступа к переменным этого типа можно использовать все внутреннее адресное пространство (256 байт)
bdata	Побитно адресуемая внутренняя память данных. Она поддерживает смешанный побитовый и побайтовый доступ
xdata	Внешняя память данных (64 кбайт). Доступна через команды MOVX @DPTR

Спецификатор типа памяти может включаться в объявление переменной наряду с ее типом, например:

```
char data var1;
char code text[] = “ENTER PARAMETER:”;
unsigned long xdata array[100];
float idata x,y,z;
unsigned char xdata vector[10][4][4];
char bdata flags;
```

Если для переменной не задан тип памяти, компилятор неявно размещает ее в пространстве памяти, которое определяется по умолчанию моделью памяти: small или large. Аргументы функций и автоматические переменные, которые невозможно разместить в регистрах, также сохраняются в области памяти по умолчанию.

Типы данных

Наряду со стандартными типами данных в компиляторах C для 8051 также определены специфичные типы, которые можно использовать в коде на C. В таблице 7 показаны некоторые из них.

Таблица 7. Дополнительные типы данных компилятора Cx51

Типы данных	Количество бит	Количество байт	Диапазон значений
bit	1		0 или 1
sbit	1		0 или 1
sfr	8	1	От 0 до 255
sfr16	16	2	От 0 до 65 535

Заметим, что типы данных имеют размеры, стандартные для компиляторов C 8051. Используются размеры, приведенные в таблице 8.

Таблица 8. Размеры стандартных типов данных в компиляторах C для 8051

Тип данных	Размер, бит
char	8
int	16
long	32
float	32
double	64

Указатели

Из-за особенностей архитектуры 8051 работать с указателями сложно. Например, адрес переменной во внутренней памяти данных — восьмиразрядный, поэтому указатель на переменную в этой памяти тоже восьмиразрядный. В то же время указатель на переменную во внешней памяти данных или программ — 16-разрядный.

Указатели на память конкретного типа

В объявлении указателя на память конкретного типа (Memory-specific pointers) всегда приводится тип памяти, и такой указатель всегда соответствует определенной области памяти, например:

```
char data *str;         /* указатель на внутреннюю память данных */
int xdata *numtab;  /* указатель на внешнюю память данных */
long code *powtab;  /* указатель на память программ */
```

Указатели на память конкретного типа могут занимать один байт (указатели на idata, data, bdata) или два байта (указатели на code или xdata).

Универсальные указатели

Компиляторы C для 8051 позволяют использовать универсальные указатели (Generic pointers). Они объявляются, как в стандартном C:

```
char *s;        /* указатель на строку */
int *numptr;  /* указатель на целое */
```

Для хранения универсальных указателей всегда используются три байта. Первый байт указывает на тип памяти, второй содержит старший байт смещения, а третий — младший байт смещения. Универсальные указатели можно применять для доступа к любым переменным, независимо от их принадлежности к областям памяти 8051. Код, в котором есть универсальные указатели, работает медленнее, и объем его больше из-за требуемых преобразований и необходимости связи с другими библиотеками. Однако все это может быть оправдано, если нужно одновременно работать с разными областями памяти.

Приведенной здесь информации нам хватит для того, чтобы начать писать программы для IP-ядра 8051s.

Литература

1. Иоффе Д., Максимов А. Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi // Компоненты и технологии. 2014. № 1.
2. Core8051s v2.4 Handbook. Материал с сайта корпорации Microsemi.
3. Core8051 Instruction Set Details User’s Guide. Материал с сайта корпорации Microsemi.