3 РАБОЧИЕ РЕЖИМЫ И СОСТОЯНИЯ

Процессор поддерживает следующие три режима работы:
- Пользовательский режим
- Режим Супервизора
- Режим Эмуляции

В режимах Эмуляции и Супервизора доступ к ресурсам ядра не ограничен. В пользовательском режиме доступ к определенным ресурсам системы ограничен, что позволяет организовать защищённую программную среду.

В пользовательском режиме обычно выполняются прикладные программы. Режим Супервизора и режим Эмуляции обычно зарезервированы под код ядра операционной системы.

Режим работы процессора определяется Контроллером событий. При обслуживании прерывания, немаскируемого прерывания (NMI) или исключения процессор находится в режиме Супервизора. При обслуживании эмуляции процессор находится в режиме Эмуляции. Когда процессор не производит обслуживания событий, он находится в Пользовательском режиме.
Текущий режим работы процессора может быть определен путём опроса регистра `IPEND`, отображенного в карте памяти, в соответствии с таблицей 3-1.

Чтение регистров, отображенных в карте памяти, в пользовательском режиме невозможно.

Таблица 3-1. Определение текущего режима работы процессора.

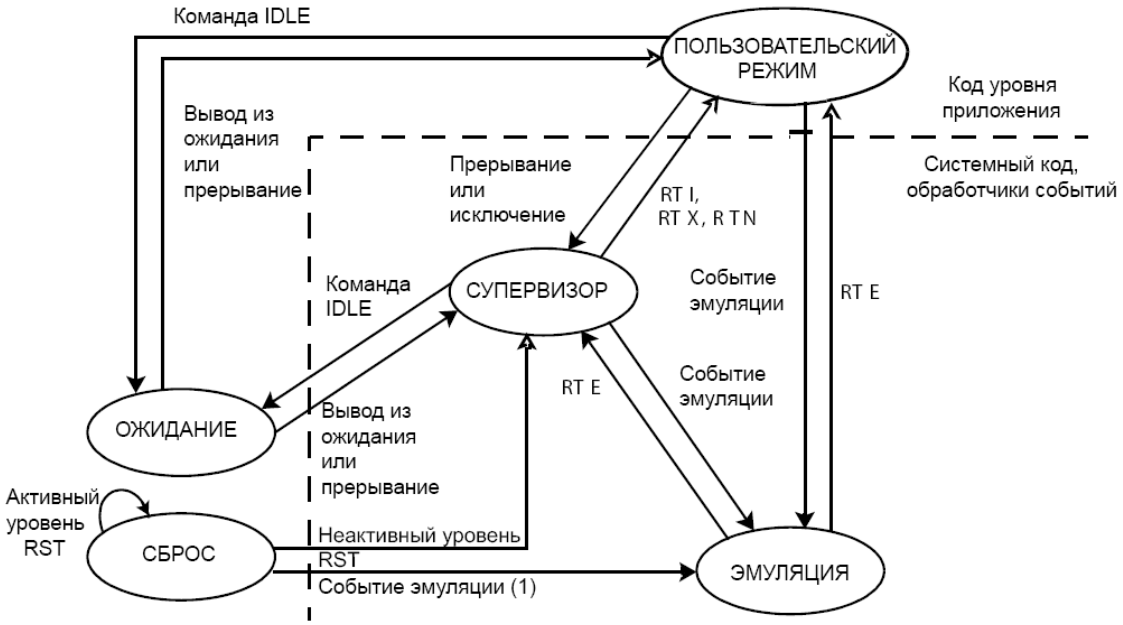
Событие	Режим	IPEND
Прерывание	Супервизор	≥ 0x10 биты IPEND[0], IPEND[1], IPEND[2] и IPEND[3]=0
Исключение	Супервизор	≥0x08 Ядро обрабатывает исключения, если IPEND[0]=0, IPEND[1]=0, IPEND[2]=0, IPEND[3]=1. При этом биты IPEND[15:4] могут содержать произвольные значения.
Немаскируемое прерывание	Супервизор	≥0x04 Ядро обрабатывает немаскируемое прерывание, если IPEND[0]=0, IPEND[1]=0, IPEND[2]=1. При этом биты IPEND[15:3] могут содержать произвольные значения.
Сброс	Супервизор	=0x02 При выходе из состояния сброса в регистр IPEND записывается значение 0x02, программа, расположенная по вектору сброса, работает в режиме Супервизора.
Эмуляция	Эмулятор	=0x01 Процессор находится в режиме эмуляции, если IPEND [0]=1 вне зависимости от состояния остальных битов IPEND[15:1].
Обслуживание событий не производится	Пользовательский	=0x00

Рабочие режимы и состояния

В дополнение к трем рабочим режимам процессор поддерживает два невычислительных состояния:

- состояние ожидания (Idle);
- состояние сброса

На рис. 3-1 показаны режимы и состояния процессора и условия переходов между ними.



(1) В нормальном режиме из состояния Сброса процессор выходит в режим Супервизора. Однако сброс может быть вызван аппаратным эмулятором. В таком случае из состояния Сброса процессор выходит в режим Эмуляции.

Рис. 3-1 Режимы и состояния процессора.

Пользовательский режим

Процессор находится в Пользовательском режиме, если он не находится в состоянии Сброса или Ожидания и, если он не обслуживает прерывание, немаскируемое прерывание, исключение или эмуляцию. Пользовательский режим используется для обработки кода уровня приложений, не требующего явного доступа к регистрам системы. Любая попытка обращения к регистрам системы, доступ к которым ограничен, вызывает исключение. В таблице 3-2 перечислены регистры, доступные в Пользовательском режиме.

Таблица 3-2. Регистры, доступные в пользовательском режиме.

Регистры процессора	Названия регистров
Регистры данных	R[7:0], A[1:0]
Регистры указателей	P[5:0], SP, FP, I[3:0], M[3:0], L[3:0], B[3:0]
Регистры состояния и регистры программного автомата	RETS, LC[1:0], LT[1:0], LB[1:0], ASTAT, CYCLES, CYCLES2

Защищенные ресурсы и команды

Ресурсы системы состоят из поднабора регистров процессора, всех регистров, отображенных в памяти, и поднабора защищенных команд. Регистры ядра и системы, отображенные в карте памяти, расположены по адресам, начиная с 0xFFC0 0000. Эта область памяти защищена от доступа в Пользовательском режиме. Попытки доступа к регистрам, отображенным в карте памяти, в Пользовательском режиме вызывают исключение.

В таблице 3-3 приведён список защищённых команд. Попытка вызова любой защищённой команды в Пользовательском режиме вызывает исключение

Таблица 3-3. Защищенные команды.

Команда	Описание
RTI	Возврат из прерывания
RTX	Возврат из исключения
RTN	Возврат из немаскируемого прерывания
CLI	Запрещение прерываний
STI	Разрешение прерываний
RAISE	Принудительный вызов прерывания/сброса
RTE	Возврат из эмуляции Вызывает исключение только при выполнении вне режима эмуляции

Защищенная память

Дополнительные области памяти могут быть защищены от доступа в Пользовательском режиме. Для этого могут создаваться и использоваться элементы ассоциативного буфера атрибутов защиты и кэширования (Cacheability Protection Look-aside Buffer, CPLB). Дополнительную информацию см. в разделе “Устройство управления памятью” в главе 6.

Вход в пользовательский режим.

При выходе из состояния Сброса процессор переходит в режим Супервизора, так как при этом он выполняет обслуживание сброса. Для перехода из состояния Сброса в Пользовательский режим необходимо выполнить два шага. Во-первых, необходимо загрузить в регистр `RETI` адрес возврата. Во-вторых, необходимо выполнить команду `RTI`. В следующем примере приведена программа, реализующая вход в Пользовательский режим при сбросе.

Пример кода, реализующего вход в Пользовательский режим при сбросе.

В листинге 3-1 представлен код, реализующий переход из состояния Сброса в Пользовательский режим.

Листинг 3-1.

```
P1.L = START;     /* Точка начала кода, выполняемого в
Пользовательском режиме */
P1/H = START;
RETI = P1;
RTI;    /*Возврат из программы обслуживания сброса */

START:          /* Поместите здесь код, выполняемый в
Пользовательском режиме */
```

Команды возврата, активизирующие пользовательский режим.

В таблице 3-4 приведен краткий обзор команд возврата, которые могут использоваться для активизации Пользовательского режима из программ обслуживания различных событий процессора. При использовании этих команд в программах обслуживания, необходимо предварительно загрузить адрес возврата в регистр `RETx` соответствующего события. Для прерывания, программа обслуживания которого может быть прервана другим прерыванием, адрес возврата помещается в стек. В данном случае, восстановление адреса осуществляется путем его извлечения из стека в регистр `RETI`. После загрузки регистра `RETI` может быть вызвана команда `RTI`.

Следует отметить, что извлечение из стека не является обязательным. Если содержимое регистра `RETI` не помещается в стек и не извлекается из стека, программа обслуживания прерывания не должна прерываться.

Процессор остается в Пользовательском режиме до тех пор, пока не произойдет одно из следующих событий:

- активизация режима Супервизора прерыванием, немаскируемым прерыванием или исключением;
- активизация режима Эмуляции при выполнении эмуляции;
- активизация состояния Сброса при выполнении сброса.

Таблица 3-4. Команды возврата, которые могут активизировать пользовательский режим.

Выполняемый процесс	**Используемая команда возврата**	**Регистр, содержащий адрес, по которому возобновляется выполнение программы**
Программа обслуживания прерывания	RTI	RETI
Программа обслуживания Исключения	RTX	RETX
Программа обслуживания немаскируемого прерывания	RTN	RETN
Программа обслуживания Эмуляции	RTE	RETE

Режим Супервизора

Процессор обслуживает все прерывания, немаскируемые прерывания и исключения в режиме Супервизора.

Режим Супервизора имеет полный неограниченный доступ ко всем ресурсам системы процессора, включая ресурсы эмуляции, при условии, что не используется CPLB, сконфигурированный соответствующим образом. Более подробную информацию см. в разделе «Устройство Управления Памятью» в главе 6. Только в режиме Супервизора возможно использование альтернативного имени регистра `USP`, определяющего положение в памяти Указателя Пользовательского Стека. Использование этого альтернативного имени необходимо, так как в режиме Супервизора по имени `SP` выполняется обращение к указателю стека ядра операционной системы, а не к указателю пользовательского стека.

Нормальная работа процессора начинается при переходе из состояния Сброса в режим Супервизора по снятию активного уровня сигнала $\overline{\texttt{RESET}}$. Процессор находится в режиме Супервизора до тех пор, пока эмуляция или вызов команды возврата не изменят режим работы. Перед вызовом команды возврата в регистр `RETI` необходимо загрузить достоверный адрес возврата.

Работа в среде без операционной системы

При работе в среде, в которой не используется операционная система, код прикладной программы должен выполняться в режиме Супервизора, чтобы сохранить возможность доступа ко всем ресурсам ядра и системы. При снятии активного уровня сигнала $\overline{\texttt{RESET}}$ процессор инициирует начало работы, обслуживая сброс. Единственным событием, способным прервать этот процесс, является эмуляция. Следовательно, обработка событий с более низким приоритетом при этом невозможна.

Одним из способов сохранить процессор в режиме Супервизора и при этом позволить обработку прерываний с низким приоритетом может являться настройка и принудительный вызов прерывания с низшим приоритетом (`IVG15`). События и прерывания описываются в разделе «События и управление событиями» в главе 4. После вызова командой `RAISE 15` прерывания с низким приоритетом, в регистр `RETI` можно загрузить адрес возврата, указывающий на пользовательский код, который может выполняться до вызова `IVG15`. После загрузки `RETI` для возврата из сброса может быть вызвана команда `RTI`.

Обработчик прерывания `IVG15` может настраиваться на выполнение перехода к начальному адресу кода приложения. Использование дополнительной команды RTI не требуется. В результате, процессор остается в режиме Супервизора, так как установлен бит `IPEND[15]`. В данный момент процессор обслуживает прерывание с низшим приоритетом, что гарантирует возможность обработки прерываний с более высоким приоритетом.

Пример кода, реализующего сохранение режима Супервизора при выходе из состояния Сброса

Для того, чтобы процессор оставался в режиме Супервизора при выходе из состояния Сброса используйте код, приведённый в листинге 3-2.

Листинг 3-2.

```
P0.L = L0(EVT15);  /* Указатель на вектор IVG15 в таблице
векторов событий */
P0.H = HI(EVT15);
P1.L = START;   /* Указатель на начало пользовательского
кода */
P1.H = START;
[P0] = P1;       /* Адрес начала пользовательского кода
заносится в вектор IVG15 таблицы векторов событий */

P0.L = L0(IMASK);
R0 = [P0];
R1.L = EVT_IVG15 & 0xFFFF;
R0 = R0|R1;
[P0] = R0;     /* В регистре маскирования прерываний
устанавливается бит, разрешающий прерывание IVG15 */

RAISE 15;   /* Вызов прерывания IVG15 */
P0.L = WAIT_HERE;
P0.H = WAIT_HERE;
RETI = P0;  /* В регистр RETI загружается адрес возврата */
RTI;        /* Возврат из события Сброса */
WAIT_HERE:  /* Ожидание обслуживания прерывания IVG15 */

JUMP WAIT_HERE;

START:            /* По вектору прерывания IVG15 программа
переходит в эту точку */
[--SP] = RETI;      /* Разрешение прерываний и сохранение
адреса возврата в стек */
```

Режим Эмуляции

Процессор входит в режим Эмуляции, если этот режим разрешён и выполняется одно из следующих условий:

- возникновение внешней эмуляции;
- вызов команды EMUEXCPT.

Процессор остается в режиме Эмуляции до выполнения программой обслуживания эмуляции команды RTE. Если при выполнении команды RTE отсутствуют запросы прерываний, ожидающих обслуживания, процессор

переключается в Пользовательский режим. В противном случае, процессор переключается в режим Супервизора для обслуживания прерывания.

Режим Эмуляции имеет высший приоритет. В данном режиме процессор имеет неограниченный доступ ко всем ресурсам системы.

Состояние Ожидания

В состоянии ожидания по желанию пользователя приостанавливается деятельность процессора. Обычно это состояние используется для снижения потребляемой мощности в течение пауз в работе процессора. При нахождении в состоянии Ожидания процессор не выполняет обработку данных. Состояние Ожидания активизируется последовательной командой `IDLE`. Команда `IDLE` сигнализирует аппаратным средствам процессора о запросе состояния Ожидания. По команде `SSYNC` ядро и внешняя система прекращают выполнение всех запланированных действий и переходных процессов.

Процессор остается в состоянии Ожидания до тех пор, пока периферийное или внешнее устройство, такое как последовательный порт или часы реального времени, не сгенерирует прерывание, требующее обслуживания.

В примере, приведённом в листинге 3-3, запрещаются прерывания ядра и выполняется команда `IDLE`. По завершении обслуживания всех процессов, запрещается подача тактовых сигналов ядра. Так как прерывания запрещены, выход из состояния Ожидания возможен только по активному уровню сигнала `WAKEUP`. Дополнительную информацию см. в разделе “Регистр разрешения вывода из ожидания по прерыванию системы (SIC_IWR)” в главе 4. (При возникновении сигнала `WAKEUP` также возможно, но не является обязательным, разрешение прерываний).

При установлении активного уровня сигнала `WAKEUP` процессор “пробуждается” (выходит из ожидания), и по команде `STI` разрешаются прерывания.

Пример кода, реализующего переход в состояние Ожидания

Для осуществления перехода в состояние Ожидания используйте код, представленный в листинге 3-3.
Процессор может выполнять необязательное отключение тактовых сигналов памяти в состоянии Ожидания. Дополнительную информацию см. в разделе “Регистр конфигурации системы (SYSCFG)” в главе 4.

Листинг 3-3.

```
CLI R0;       /* Запрещение прерываний */
IDLE;         /*Очистка конвейера и перевод ядра в состояние
Ожидания */
STI  R0;         /*  Разрешение  прерываний  после  выхода  из
ожидания */
```

Состояние Сброса

В состоянии Сброса выполняется инициализация логики процессора. В этом состоянии код прикладных программ и операционной системы не выполняется, тактирование процессора останавливается.

Процессор остаётся в состоянии Сброса, пока внешняя логика удерживает активный уровень внешнего сигнала RESET. При снятии активного уровня сигнала RESET процессор завершает процедуру сброса и переходит в режим Супервизора, в котором производится выполнение кода по адресу, определяемому вектором сброса.

Программа, выполняющаяся в режиме Супервизора или Эмуляции, может активизировать состояние Сброса без использования внешнего сигнала RESET, вызывая соответствующую версию команды RAISE.

Прикладные программы, выполняемые в Пользовательском режиме, не могут активизировать состояние Сброса. Исключением является использование системных вызовов, предоставляемых ядром операционной системы. В таблице 3-5 приведена краткая сводка по состоянию процессора при сбросе.

Таблица 3-5. Состояние процессора при сбросе.

Параметр	Описание состояния при сбросе
Ядро	
Рабочий режим	Режим Супервизора, событие сброса. Тактирование приостановлено
Режим округления	Несмещенное округление
Счетчики тактов	Запрещены, содержимое равно нулю
Регистры DAG (I,L,B,M)	Случайные значения (при инициализации должны быть сброшены)
Адресные регистры и регистры данных	Случайные значения (при инициализации должны быть сброшены)
IPEND, IMASK, ILAT	Сброшены, прерывания глобально запрещены битом 4 регистра IPEND
CPLB	Не используется
Память команд L1	SRAM (использование в качестве кэша запрещено)
Память данных L1	SRAM (использование в качестве кэша запрещено)
Биты достоверности кэша	Соответствуют недостоверности кэша
Система	
Методы загрузки	Определяются состоянием выводов BMODE при сбросе
Тактовая частота MSEL	Значение при сбросе = 10
Режим обхода PLL	Запрещён
Отношение частоты VCO к частоте ядра	Значение при сбросе = 1
Отношение частоты VCO к частоте системы	Значение при сбросе = 5
Тактирование периферии	Запрещено.

Сброс системы и конфигурация при включении питания

В таблице 3-6 описаны пять типов сброса. Необходимо отметить, что все типы сброса, за исключением программного сброса системы, приводят к сбросу ядра.

Таблица 3-6. Типы сброса.

Тип сброса	Источник	Результат
Аппаратный сброс	Аппаратный сброс вызывается сигналом на выводе $\overline{\text{RESET}}$	Выполняется сброс ядра и периферии, включая контроллер динамического управления питанием (DPMC). Сбрасывается бит отсутствия загрузки при программном сбросе в регистре SYSCR. Дополнительную информацию см. в разделе «Регистр конфигурации сброса системы».
Программный сброс системы	Программный сброс системы вызывается записью b#111 в биты [2:0] регистра SWRST, отображенного в карте памяти по адресу 0xFFC00410.	Выполняется только сброс периферии, за исключением блока RTC (часов реального времени) и большей части DPMC. DPMC сбрасывает только бит отсутствия загрузки при программном сбросе в регистре SYSCR. Сброс ядра не производится. Процедура загрузки не инициируется.
Сброс по сторожевому таймеру	Сброс вызывается при соответствующем программировании сторожевого таймера	Выполняется сброс ядра и периферии, за исключением блока RTC и большей части DPMC. Определить, являлся ли источником сброса сторожевой таймер, можно путём чтения регистра программного сброса (SWRST).
Сброс при двойной ошибке ядра	Сброс при входе ядра в состояние двойной ошибки может быть вызван снятием бита маскирования сброса при двойной ошибке ядра в регистре маскирования прерываний контроллера прерываний системы (SIC_IMASK)	Выполняется сброс ядра и периферии, за исключением блока RTC и большей части DPMC. Определить, являлась ли источником сброса двойная ошибка ядра, можно путём чтения регистра SWRST.
Программный сброс ядра	Программный сброс ядра вызывается командой RAISE 1 или установкой программой эмуляции по порту JTAG бита программного сброса в регистре управления отладкой ядра (DBGCTL). Регистр DBGCTL не отображён в карте памяти.	Выполняется только сброс ядра. Периферия не распознаёт этот тип сброса.

Аппаратный сброс

Аппаратный сброс процессора является асинхронным событием. Для выполнения аппаратного сброса необходимо установить низкий уровень сигнала на входном выводе $\overline{\text{RESET}}$. Дополнительную информацию см. в *ADSP-BF531/ADSP-BF532/ADSP-BF533 Embedded Processor Data Sheet*.

Аппаратно инициированный сброс приводит к сбросу всей системы в целом, включая и ядро, и периферию. После установления активного уровня сигнала на

выводе $\overline{\texttt{RESET}}$ процессор убеждается в том, что все асинхронные периферийные устройства распознали и выполнили сброс. После сброса процессор переходит к процедуре загрузки, определяемой состоянием выводов BMODE.

Выводы BMODE[1:0] предназначены для управления режимом. Эти выводы не выполняют других функций и могут быть привязаны к V_{DD} или V_{SS} для задания постоянного значения, соответствующего определённому режиму. Выводы BMODE[1:0] и соответствующие биты регистра SYSCR определяют режим загрузки, выполняемой после аппаратного сброса или программного сброса системы. Дополнительную информацию см. в разделе “Сброс” главы 4 и таблице 4-11 “События, вызывающие исключения”.

Регистр конфигурации системы при сбросе (SYSCR)

Значения выводов BMODE[1:0] фиксируются в регистре конфигурации системы при сбросе (SYSCR) при снятии активного уровня сигнала на выводе $\overline{\texttt{RESET}}$. После выполнения процедуры аппаратного сброса значения становятся доступны для чтения или модификации программой. Программа может модифицировать только бит отсутствия загрузки при программном сбросе.

Различные конфигурационные параметры в SYSCR управляют различными настройками (см. рис. 3-2).

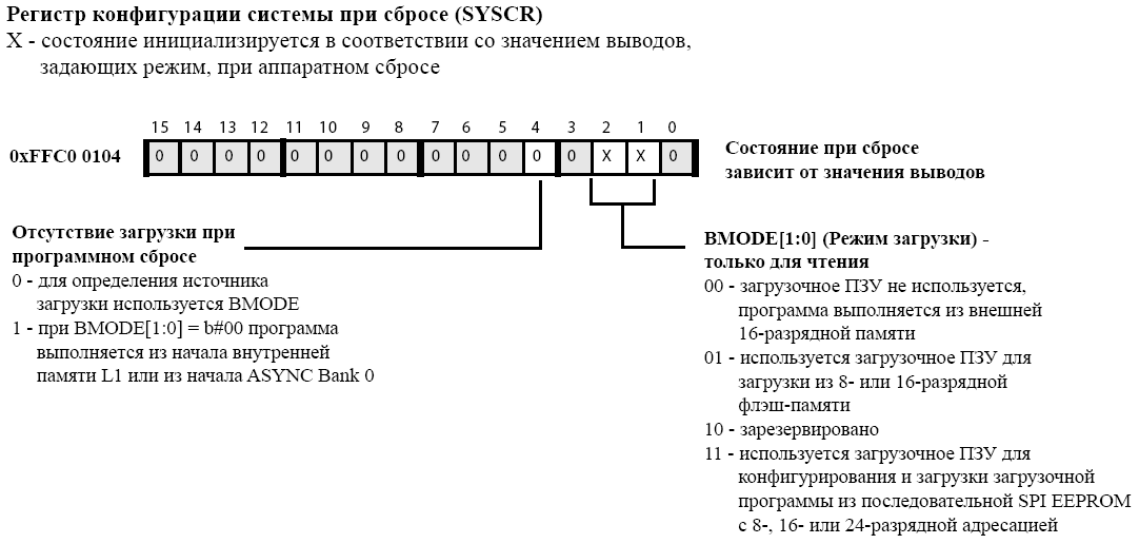


Рис. 3-2. Регистр конфигурации системы при сбросе

Типы программного сброса и сторожевой таймер

Программный сброс может инициироваться тремя способами:

- сторожевым таймером, сконфигурированным соответствующим образом;
- установкой поля программного сброса системы в регистре программного сброса (см. рис. 3-3);
- при помощи команды RAISE 1.

Сторожевой таймер выполняет сброс и ядра, и периферии. Программный сброс системы приводит к сбросу периферии, при этом не производится сброс ядра и не инициируется процедура загрузки.

ⓘ Программный сброс системы должен производиться при выполнении программ из памяти L1 (сконфигурированной либо как кэш, либо как SRAM).

Если память команд L1 сконфигурирована как кэш, следует убедиться в том, что последовательность программного сброса системы была считана в кэш.

После инициации сброса сторожевым таймером или программного сброса системы, процессор убеждается в том, что все асинхронные периферийные устройства распознали и выполнили сброс.

При сбросе, вызванном сторожевым таймером, процессор переходит к процедуре загрузки. Режим загрузки конфигурируется состоянием битов BMODE и бита отсутствия загрузки при программном сбросе.

Если бит отсутствия загрузки при программном сбросе в регистре SYSCR сброшен, процедура загрузки определяется битами управления BMODE[1:0].

Регистр программного сброса (SWRST)

Программный сброс может инициироваться установкой поля Программного Сброса Системы в регистре Программного Сброса (SWRST). Бит 15 сигнализирует о том, происходил ли программный сброс после последнего чтения регистра SWRST. Биты 14 и 13 указывают, являлся ли источником программного сброса программный сторожевой таймер или двойная ошибка ядра, соответственно. Биты [15:13] доступны только для чтения, при чтении регистра они сбрасываются. Биты [3:0] доступны для чтения и записи.

Когда значение выводов BMODE не равно b#00, и в регистре SYSCR установлен бит отсутствия загрузки при программном сбросе, процессор начинает выполнение программы из начала внутренней памяти L1. В данной конфигурации ядро начинает выборку команд из начала внутренней памяти L1.

Когда значение контактов BMODE равно b#00, ядро начинает выборку команд с адреса 0x2000 0000 (начало банка ASYNC0).

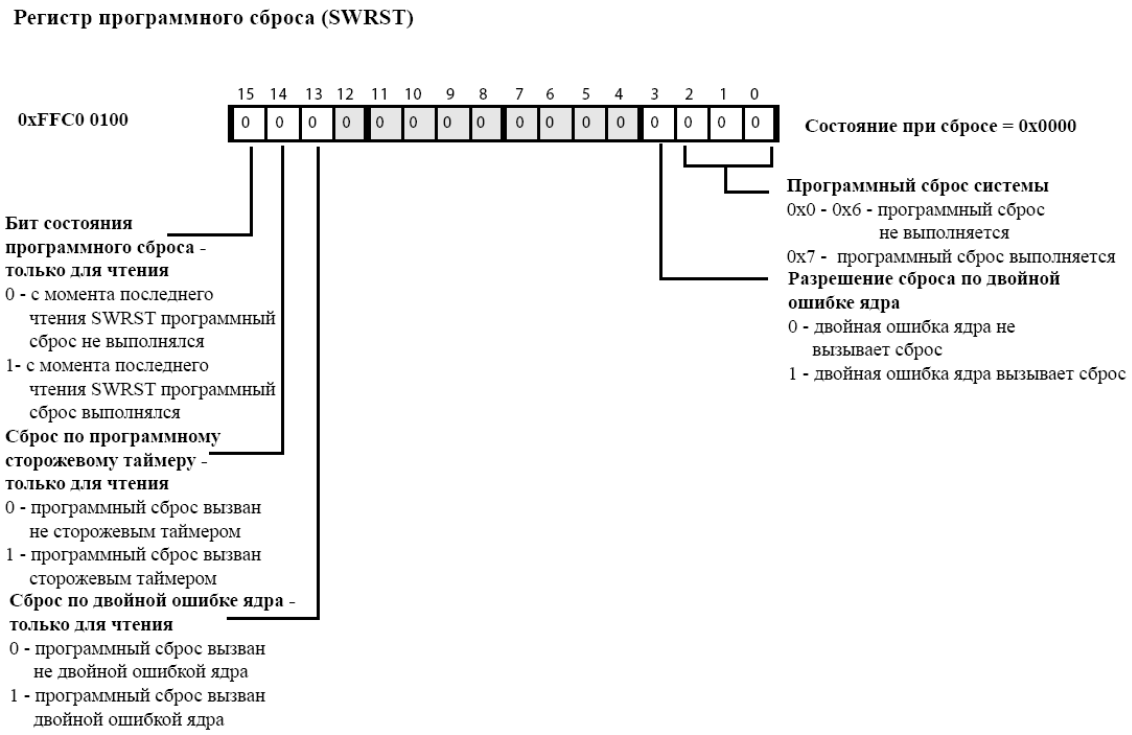


Рис. 3-3. Регистр программного сброса.

Программный сброс ядра

Программный сброс ядра инициируется выполнением команды RAISE 1 или установкой программой эмуляции по порту JTAG бита программного сброса (SYSRST) в регистре управления отладкой ядра (DBGCTL). Регистр DBGCTL не отображается в карте памяти.

Программный сброс ядра влияет только на состояние ядра. Следует отметить, что в зависимости от действий, выполняемых ядром в момент сброса, ресурсы системы могут принимать неопределенное, или даже недостоверное состояние.

Сброс ядра и системы

Для выполнения сброса ядра и системы используйте последовательность команд, представленную в листинге 3-4.

Листинг 3-4.

```
/* Вызов программного сброса */
 P0.L = L0(SWRST);
P0.H = HI(SWRST);
R0.L = 0x0007;
W[P0] = R0;
```

```
SSYNC;

/* Снятие программного сброса */
P0.L = L0(SWRST);
P0.H = HI(SWRST);
R0.L = 0x0000;
W[P0] = R0;
SSYNC;

/* Сброс ядра – вызывает перезагрузку */
RAISE 1;
```

Методы загрузки

Внутреннее загрузочное ПЗУ содержит небольшое загрузочное ядро, которое можно либо не использовать, либо использовать для загрузки пользовательского кода из внешней памяти. Дополнительную информацию см. в таблице 4-10 “Адреса вектора сброса”. Загрузочное ядро выполняет чтение состояния выводов `BMODE[1:0]` в момент сброса, идентифицируя источник загружаемого кода (см. таблицу 4-7). При работе в режиме загрузки 0 процессор настраивается на выполнение программы из 16-разрядной внешней памяти по адресу 0x2000 000 (Банк ASYNC0).

Существует несколько режимов загрузки пользовательского кода из внешней памяти. В этих режимах загрузочное ядро производит настройку выбранного периферийного устройства в соответствии со значениями выводов `BMODE[1:0]`.

В каждом из режимов загрузки пользовательский код, считываемый из внешней памяти, помещается в начало памяти L1. В соответствии с информацией, содержащейся в заголовке файла загрузчика, может производиться чтение дополнительных разделов во внутреннюю память. Загрузочное ядро прекращает процесс загрузки при выполнении перехода к началу пространства памяти команд L1. Затем процесс начинает выполнение программы с этого адреса.

При загрузке по SPI в качестве источника сигнала выбора микросхемы SPI используется вывод 2 флага общего назначения. Для корректного выполнения операции необходимо соединить данный вывод соответствующим образом.

При программном сбросе ядра также осуществляется переход ядра в соответствии с вектором, определяющим загрузочное ПЗУ. При этом типе сброса выполняется только сброс ядра; он не воздействует на остальную часть системы. Для того, чтобы предотвратить нежелательную инициацию загрузки, ядро, содержащееся в загрузочном ПЗУ, считывает бит отсутствия загрузки при программном сбросе в регистре `SYSCR`. Если при программном сбросе этот бит установлен, процессор пропускает стандартную процедуру загрузки, осуществляет переход в начало памяти L1 и начинает выполнение программы.

Для нормальной работы загрузочного ядра в режиме загрузки из флэш-памяти предполагается выполнение следующих условий:

- разрешен банк 0 асинхронной памяти (AMB0);
- разрешена 16-разрядная упаковка слов для AMB0;
- сигнал RDY AMB0 установлен в активный высокий уровень;
- время удержания банка 0 (от снятия сигнала чтения/записи до снятия сигнала $\overline{\texttt{AOE}}$) = 3 такта;
- времена доступа чтения/записи банка 0 = 15 тактов.

Программа загрузочного ядра предполагает использование скорости передачи SPI, равной 500 кГц. Осуществляется поддержка SPI-совместимых последовательных EEPROM с 8-, 16- и 24-разрядной адресацией. Для выбора SPI-совместимого устройства EEPROM порт SPI использует выходной вывод `PF2`. Контроллер SPI последовательно выдаёт команды чтения по адресам 0x00, 0x0000 и 0x000000, пока не обнаружит наличие EEPROM с 8-, 16- или 24-разрядной адресацией. Затем начинается синхронная передача данных из EEPROM в начало памяти команд L1.

В каждом из режимов загрузки сначала выполняется чтение из внешнего устройства памяти 10-байтного заголовка. Он определяет число передаваемых байтов и адреса в памяти, в которые будет осуществляться передача. В течение одной процедуры загрузки может быть загружено несколько блоков памяти. По окончании загрузки всех блоков начинается выполнение программы из начала SRAM команд L1.

В устройствах, имеющих внутреннее ПЗУ, загрузочное ПЗУ никогда не применяется, и выполнение программы начинается с начального адреса внутреннего ПЗУ (0xFFA0 0000).