СПИСОК АППАРАТНЫХ ОШИБОК И ЗАМЕЧАНИЙ ПО ПРИМЕНЕНИЮ

Мультиклеточный процессор MultiClet R1

Мультиклеточные процессоры:

MCp042R100102-LQ256

Содержание

1 Уровни ошибок

Ошибки в процессоре отсортированы на 3 уровня критичности:

Уровень 1: Ошибочное поведение, которое невозможно обойти. Ошибки данного уровня серьезно ограничивают использование продукта.

Уровень 2: Ошибочное поведение, которое ограничивает часть целевых функций, но при этом продукт является пригодным для большинства приложений.

Уровень 3: Ошибочное поведение, которое не было изначально определено, но не вызывает проблем при соблюдении рекомендаций.

Примечание: все аппаратные ошибки будут исправлены в следующей версии процессора. Большинство аппаратных ошибок обходится на уровне компилятора Си.

2 Ошибки уровня 1

Ошибок первого уровня не зафиксировано.

3 Ошибки уровня 2

3.1 (201) Искажение кодов инструкций при переходе между областями памяти

Описание: при определённых условиях в процессе выборки может происходить искажение инструкций, приводящее к неправильному выполнению программы и возникновению немаскируемого программного исключения.

Условие: если в программе

```
para1:  jmp para2
        ...
        complete

para2:  ...
        complete
```

параграфы `para1` и `para2` попадают в разные области памяти (например, первый — в память программ, а второй — в память данных, либо в память программ и внешнюю память, соответственно). Кроме того, данная ошибка может произойти при обработке прерываний, если обработчик прерывания и прерываемая программа находятся в разных областях памяти: искажение инструкций может произойти как при входе в обработчик прерывания, так и при возврате в основную программу.

Также уязвимой по отношению к данной ошибке является следующая ситуация:

```
para1:  ...
        complete

para2:  jmp para1
        ...
        instrx
        complete
```

Даже если `para1` и `para2` размещаются в одной области памяти, но последняя инструкция `instrx` находится вблизи (восемь 32-битных слов и ближе) границы области памяти, параграф `para1` будет искажён.

Рекомендации и способы обхода:

1. Избегать размещения исполняемого кода в нескольких областях памяти, а также ближе восьми 32-битных слов к нижней границе области памяти. Если это невозможно, использовать программную "подкачку"кода (т.е. перед выполнением копировать необходимый блок кода в текущую область памяти, например, с помощью DTC) и оверлеи.

2. Если выполнение п. 1 невозможно, необходимо модифицировать *все* параграфы, *на* которые может передаваться управление после пересечения границы областей памяти, добавив в их начало следующую преамбулу:

```
.rept 8
        getl @0
.endr
```

   Т.е. для правильной работы кода из первого примера (два связанных параграфа находятся в разных областях памяти) он должен выглядеть следующим образом:

```
para1:  jmp para2
        ...
        complete

para2:
.rept 8
        getl @0
.endr
        ...
        complete
```

3.2 (202) Из I^2S не читаются принимаемые данные

Описание: Из регистра `I2S_DATA` нет возможности вычитать принимаемые данные. Тактирование и передача работают корректно.

Условие: при чтении данных из регистра `I2S_DATA`.

Рекомендации и способы обхода: реализовать прием данных путем прерываний по линии GPIO.

3.3 (203) Прерывание для трёх арифметических операций

Описание: если выполняется операция целочисленного умножения, деления двойной точности или квадратный корень, то результат операции будет потерян при возникновении прерывания, и процессор зависает.

Условие: при возникновении прерывания в момент выполнения целочисленного умножения, деления двойной точности, и любого квадратного корня.

Рекомендации и способы обхода: использовать восьмой бит `PSW` на участках, где задействованы данные команды, и может сработать прерывание.

Компилятор Си автоматические использует обход в генерируемом коде, если при компиляции была указана опция `-Wf-fix-imul`.

3.4 (204) Блокирование работы блока доступа в память (DMS) на чтение

Описание: при сочетании определённых условий, возникновение прерывания может привести к блокированию работы блока доступа в память (DMS) на чтение; что приводит к последующей невозможности выполнять инструкции, зависящие от результатов заблокированных чтений, и, в конечном итоге, к зависанию процессора.

Условие: при выдаче на исполнение в блок доступа в память (DMS) двух команд *подряд*, при одновременной занятости шины результатов клетки более приоритетным блоком, и поступлении в этот же момент запроса на прерывание. Наиболее вероятно возникновение данной ситуации при выполнении кода следующего типа *одной* клеткой на фоне прерываний:

```
para:  rdsl addr1
       getl v1
       rdsl addr2
       getl v2
       ...
       complete
```

В данном примере более приоритетным относительно блока доступа в память (DMS), выполняющего команды `rdsl`, оказывается блок управления (CU), выполняющий команды `getl`.

Рекомендации и способы обхода: не использовать подобных конструкций при выполнении программы *одной* клеткой.

4 Ошибки уровня 3

4.1 (301) Конвертирование long во float или double и обратно

Описание: при использовании команды конвертирования переменной типа long во float или double и обратно, может произойти немаскируемое прерывание `PRGE(FNAN)`, но результат команды будет верным.

Условие: если преобразуемое значение близко к минимальному или максимальному значению переменной определенного типа.

Рекомендации и способы обхода: немаскируемое прерывание не сработает (только зафиксируется в регистре `INTR`), если прерывания запрещены в системном регистре `PSW`. Если прерывания разрешены и произошло прерывание `PRGE`, то его можно обработать на основании значения в системном регистре `ER`. Однако, в последнем случае следует учитывать, что при возникновении прерывания происходит откат параграфа, и после обработки прерывания выполнение параграфа (вместе с командой, вызвавшей прерывание) повторяется, что может привести к зацикливанию программы.

4.2 (302) Соответствие типа и очередности косвенной адресации

Описание: при выполнении операции `xxx [v]`, чтение по косвенному адресу выполняется, согласно типу родительской операции. Например, при выполнении операции `subb @1,[v]` чтение из адреса `v` будет байтовое.
При выполнении `xxx [v]`, чтение по косвенному адресу выполняется аналогично инструкции `rdd`, т.е. не подчиняется контролю очерёдности чтения-записи.

Условие: при использовании косвенной адресации в операциях, имеющих типы (знаковые и беззнаковые): byte, short, float, double, complex, происходит операция чтения значения по косвенной адресации будет выполнена некорректно. При выполнении чтения по косвенной адресации не произойдет контроля очередности чтения-записи.

Рекомендации и способы обхода: не использовать косвенную адресацию для операций имеющих тип: byte, short, float, double, complex. Проконтролировать, что нет пересечений при чтении и записи в память по косвенной адресации. Использовать восьмой бит регистра PSW в параграфах, где используется косвенная адресация или добавить дополнительный промежуточный параграф с операцией чтения либо заменить косвенную адресацию на операции чтения с контролем очередности чтения-записи.

Компилятор Си обходит данную ошибку автоматически.

4.3 (303) Чтение по косвенному адресу с непредсказуемым типом

Описание: при выполнении операции перехода `jxx [@x+v]` чтение по косвенному адресу выполняется с непредсказуемым типом. Такую конструкцию использовать нельзя. Операции `jxx [#r+v]` и `jxx [v]` работают правильно.

Условие: при использовании операции перехода `jxx [@x+v]`.

Рекомендации и способы обхода: не использовать команду перехода вида `jxx [@x+v]`.

Компилятор Си обходит данную ошибку автоматически.

4.4 (304) Команды `rdd/wrd` с плавающей точкой

Описание: команды `rdd/wrd` с плавающей точкой выполняются некорректно.

Условие: при использовании команд внеочередного чтения и записи в форматах: float, double, complex.

Рекомендации и способы обхода: использовать следующие замены:

`rddf/wrdf` → `rddl/wrdl`

`rddd/wrdd` → `rddq/wrdq`

`rddc/wrdc` → `rddpl/wrdpl`

4.5 (305) Прерывание от АЦП

Описание: прерывания от АЦП происходят по фронту LRCK, т.е. когда АЦП меняет данные, поэтому чтение может давать некорректные результаты.

Условие: при работе с прерываниями от АЦП и считывании данных.

Рекомендации и способы обхода: Гарантировать, что чтение данных из АЦП произойдёт не ранее, чем через 8 периодов частоты тактирования АЦП. При необходимости реализовать задержку с использованием прерывания от таймера периферии или системного таймера.

4.6 (306) Направление линий GPIO при работе в режиме альтернативных функций

Описание: Если линия порта GPIO сконфигурирована на *выход*, то она не сможет работать, как *вход* альтернативной функции (независимо от логики работы самой альтернативной функции).

Условие: при одновременной установке '1'в битах регистров `DIR` и `BPS`.

Рекомендации и способы обхода: всегда переключать линии, используемые, как альтернативные функции, на вход записью '1'в регистр `DIR` соответствующего блока GPIO.

4.7 (307) Блокирование прерываний при выполнении очень коротких параграфов

Описание: если одна из групп клеток циклически выполняет очень короткий параграф, содержащий запись в память, например:

```
p:  jmp p
    wrl @0, xxx
    complete
```

то обработка прерываний в других группах клеток, выполняющих параллельные задачи, будет затруднена, либо невозможна вовсе.

Условие: при выполнении одной из групп клеток кода, непрерывно осуществляющего запись в память.

Рекомендации и способы обхода: избегать ситуаций, когда запись в память осуществляется непрерывно. В крайнем случае, "разбавлять" код другими операциями, например, `getl @0`.

4.8 (308) Распространение немаскируемых прерываний во все группы клеток

Описание: если при выполнении программы одной из групп клеток возникает исключительная ситуация («деление на ноль» `dz`, «неверная инструкция» `ii`, «переполнение» `ovf` или «NaN» `nan`), то немаскируемое прерывание распространяется во *все* группы клеток.

Условие: при возникновении программного исключения.

Рекомендации и способы обхода: учесть в обработчике возможность возникновения "спонтанного" немаскируемого прерывания.

4.9 (309) Блокирование операций чтения параграфами без переходов

Описание: если в программе предусмотрено "зависание" группы клеток с использованием параграфа без инструкций перехода (как это делается, например, при реконфигурации), и в таком параграфе встречаются команды записи в память:

```
stop:  getl 1
       wrl @1, flag1
       complete
```

то выполнение *всех* последующих команд чтения (исключая "прямое" чтение `rdd`) будет заблокировано во *всех* группах клеток.

Условие: при отсутствии перехода из параграфа, в котором есть команды записи в память.

Рекомендации и способы обхода: при необходимости, использовать дополнительный "пустой" параграф:

```
stop:   jmp stop2
        getl 1
        wrl @1, flag1
        complete

stop2:  getl 0
        complete
```

4.10 (310) Признак очистки буфера не всегда можно использовать для ожидания завершения выполнения всех инструкций параграфа

Описание: в некоторых случаях, не смотря на установленный признак `CLNBUF` (`PSW[8]`), возможен переход к исполнению следующего параграфа, даже если инструкции из последнего набора, декодируемого совместно с признаком `complete`, ещё не выполнились (т.е. до четырёх последних инструкций параграфа). Например, в следующем примере:

```
para1:  jmp para2
        getl 1
        getl 2
        addl @1, @2
        wrl @1, 0xDEADBEEF
        complete
para2:  getl #STOCR
        orl @1, 0x00000001
        setl #STOCR, @1
        ...
        complete
```

даже при установленном признаке `CLNBUF` (`PSW[8]`) чтение (и даже запись) в регистр `STOCR` могут произойти раньше записи в адрес `0xDEADBEEF`.

Условие: если выборка инструкций происходит настолько медленно, что, к моменту декодирования набора инструкций, содержащего признак `complete`, буфер команд оказывается пуст.

Рекомендации и способы обхода: при необходимости, использовать “пустые” инструкции в конце параграфа (по числу клеток в группе):

```
para1:  jmp para2
        getl 1
        getl 2
        addl @1, @2
        wrl @1, 0xDEADBEEF
        getl @0
        getl @0
        getl @0
```

```
        getl @0
        complete
```

Либо использовать дополнительный “пустой” параграф:

```
para1:  jmp dummy1
        getl 1
        getl 2
        addl @1, @2
        wrl @1, 0xDEADBEEF
        complete
dummy1: jmp para2
        complete
para2:  getl #STOCR
        orl @1, 0x00000001
        setl #STOCR, @1
        complete
```

Компилятор Си обходит данную ошибку автоматически.