УДК 519.688

В.С. БРЕСЛАВЕЦ, А.А. СЕРКОВ

Национальный технический университет «ХПИ», Украина

МЕТОДЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ЗАЩИЩЁННОГО ОТ ЭЛЕКТРОМАГНИТНЫХ ПОМЕХ

В статье рассматриваются основные методы, применяемые для создания программного обеспечения, защищённого от воздействия электромагнитных помех. Выполнена сравнительная характеристика этих методов и определены достоинства и недостатки каждого из них. Рассмотрена возможность применения этих методов в зависимости от типа конкретной помехи.

электромагнитная помеха, защиты данных, ЗУ, помехозащищённая конструкция, защита от сбоев, цифровой вход, прерывания, линия данных / адреса, программное обеспечение

Введение

Проблема защиты программных и аппаратных средств от сбоев, вызванных действием мощных электромагнитных помех, становится все более актуальной. Определенная защита против воздействия помех может быть предпринята при разработке программных средств. Широко используется стандартная техника сохранения данных и коррекции ошибок. Свойства аппаратуры могут быть улучшены хорошо разработанным программным средством. Они используются, когда применены оптимальные приемы создания помехозащищенной аппаратуры, и программные средства расширят возможности в этом направлении. Например, программа, которая не определяет цифровой вход до тех пор, пока троекратный опрос даст такой результат, который будет непроницаемым для импульсов, короче, чем требуется.

Постановка проблемы. Однако, если вы используете для тестирования только короткие импульсы или одиночный перепад, то аппаратура будет защищена, но длительный импульс будет вызывать сбой, который должен быть устранен помехозащищенной конструкцией. Поэтому возникает необходимость использования при проектировании программных средств таких методов, которые бы позволяли бы устранить воздействие такого рода электромагнитных помех.

Анализ литературы. Не все сбои микропроцессоров вызываются внешними помехами [1]. Другими источниками являются внутреннее соединение, предельное значение параметров конструкции, особенности функционирования программ, метастабильность асинхронных цепей и т.п. [2]. Типичной программной техникой являются [3]:

1. Контроль типов и проверка принадлежности к диапазону всех входных данных.
2. Выборка входных данных в определенные моменты времени и их усреднение для аналоговых сигналов или проверка достоверности для цифровых данных.
3. Контроль четности и контрольных сумм во всех системах передачи данных.
4. Защита блоков данных в энергозависимых запоминающих устройствах с обнаружением ошибок и коррекцией алгоритмов [4].
5. Где возможно использовать переключение относительно уровня, чем триггерное прерывание.
6. Периодическое инициализирование микросхемы программируемого интерфейса [6].

Цель статьи – анализ методов, применяемых при проектировании программного обеспечения, позволяющих предотвратить возникновение сбоев при прохождении импульсных электромагнитных помех.

Основная часть

Когда сбои определены при тестировании, часто трудно найти причину их возникновения. Подсоединение встроенных эмуляторов, подвергающихся быстрым переходным процессам, не рекомендуется. Можно рассортировать причины внутренних состояний после сбоев по состоянию портов вход/выход и шин. Если возможно выделить резервный контакт вход/выход для диагностики, то состояние системы может быть легко определено. Альтернативно или дополнительно не используемые элементы памяти могут быть предназначены для хранения диагностических результатов, которые определяются после сбоя [1]. При этом применяются следующие методы защиты программного обеспечения от сбоев:

1. Легализация и усреднение входных данных. Если возможно установить ограничение на параметры входных данных в виде цифровых сигналов путем программирования, тогда имеется возможность не допустить данные, которые не отвечают этим ограничениям. Затем, как это происходит во многих системах управления и мониторинга, каждый вход следует предназначить для своего потока данных, что является простейшим способом не ошибочных данных. Поскольку большинство видов помех представляется в виде коротких импульсов или перепадов напряжения, последовательность данных в потоке будет корректной и ничего не будет потеряно за счет игнорирования плохих входных сигналов. Приложение с определением типа данных может потребовать флаг с большей вероятностью, чем просто игнорировать их.

Эта техника может быть расширена, если известны максимальные требования к обмену данными. Вход, для которого превышены эти ограничения, может быть проигнорирован, пока данные находятся внутри этих ограничений. Программное усреднение данных в потоке приводит к выравниванию флуктуационного процесса шумов, что также способствует восстановлению данных.

При этом предполагается, что, когда используется модернизированная программа для определения ошибки, то не будут зафиксированы внутренние ошибки, которые необходимо отмечать флагом или корректировать, такие как сбои сенсора. Большинство комплексных алгоритмов необходимы для того, чтобы выявить ненормальные условия функционирования.

2. Цифровые входы. Подобный процесс проверки должен быть применен к цифровым входам. В этом случае существует только две стадии проверки, поскольку диапазон тестирования не подходящий. Примем, что входные порты будут опрошены с высокой скоростью, сравнены действительные входные данные друг с другом и не будут приняты до тех пор, пока два или три сравниваемых значения не будут в согласии. Таким образом, процессор будет «слепым» к поступающим импульсам, которые могут совпасть с опрашивающим временным слотом. (Этот метод есть вариантом стандартной техники возбуждения («de bouncing») входов.) Это требует, чтобы скорость опроса была в два или три раза быстрее, чем минимально требуемое для определения времени отклика, что может потребовать более быстрого микропроцессора, чем оригинальное изделие.

3. Прерывания. Для описанных выше случаев предпочтительно, чтобы система не зависела от прерывания по входу. Такие прерывания смогут быть вызваны короткими импульсами шумов, которые могут быть восприняты как сигнал. Сомнительные прерывания необходимы в некоторых приложениях, но параметры системы должны минимизировать их последствия. Если имеется возможность выбора, предпочтение следует отдавать уровню зависимым прерываниям по входу. Если прерывание не замаскировано, то на контак-

тах входа может появиться импульс следующих друг за другом прерываний, который даст быстрый результат в переполнении стека.

4. Эстафетная передача. Структура программного обеспечения напоминает структуру бизнес-менеджмента: большие задачи решаются наверху, но исполняются внизу [1]. Опасность приходит, когда задачи, решаемые внизу, не координированы с решаемыми на верхнем уровне. Возмущения воздействуют на микроконтроллер на низком уровне подпрограмм путем возможного нарушения этих программ, вызывая серьезные последствия. Структура «эстафетная передача» (token passing) (рис. 1) позволяет каждой подпрограмме проверять полномочия вызывающей подпрограммы. Программа верхнего уровня отправляет специальный байт (эстафету) подпрограмме первого уровня, которая проверяет его, и, если необходимо, пропускает дальше. Обычно прохождение эстафеты происходит успешно, но, если это не так, происходит перезагрузка системы. При этом существует определенная цена в стоимости, избыточности данных и времени, но выбор подобной структуры более чем глобален.

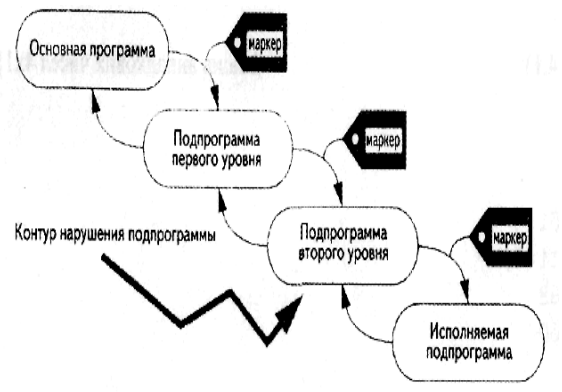

Рис. 1. Прохождение эстафеты по подпрограммам

5. Защита данных и памяти. Энергозависимая память (оперативная память, RAM, в отличие от ROM; EEPROM и флеш-память могут рассматриваться в данном случае как энергозависимые, так как их данные могут быть разрушены в момент записи) восприимчива к различным формам искажения данных. Эти искажения можно будет обнаружить, если поместить критические данные в отдельные массивы. Каждый массив необходимо будет защитить контрольной суммой и сохранить его в специальной таблице. Проверка контрольной суммы с целью диагностики может быть выполнена фоновой подпрограммой автоматически в любой момент времени, когда вы посчитаете необходимый проверить данные в оперативной памяти, после чего погрешность может быть помечена, или может быть сгенерирован программный сброс – как потребуется. Нет никакой необходимости знать абсолютные значения данных RAM при условии, что контрольная сумма будет повторно рассчитана каждый раз, когда массивы изменяются. В этом случае наиболее опасным моментом является прерывание диагностирующей подпрограммы операцией записи данных в проверяемый массив или наоборот, так как в этом случае ошибки начнут появляться из ниоткуда. Фактическое выделение разделов данных в отдельные таблицы – тоже критическое решение проектировщика системы, поскольку оно воздействует на полную устойчивость системы.

Некоторые долговременные ЗУ имеют в своем составе программный запирающий механизм, который отключает операцию записи. Полноценное использование этой особенности состоит в удержание памяти постоянно запертой, с отпиранием ее только в течение тех редких моментов, когда должны записываться новые данные. Если сбои питания или сбои микропроцессора произойдут непосредственно во время записи в долговременное запоминающее устройство, то запоминаемые данные будут неизбежно потеряны. Чтобы ликвидировать проблему, можно использовать COPY-MODIFY-STORE-ERASE – последовательность и предотвратить особо важные данные от частичной перезаписи.

6. Неиспользуемая память программы. Одна из угроз при возникновении такого рода сбоев есть возможность микропроцессора обращаться к неиспользуемому данной программой пространству памяти из-за сбоя в счетчике команд. Если он это делает, то любые данные, которые там находятся, он интерпретирует как команды программы. В таких обстоятельствах нужно, чтобы бы это действие имело предсказуемый результат.

При попытке доступа по несуществующему адресу шина возвращает сообщение $\#FF_H$, снабжаемое повышением напряжения на пассивных линиях шины. С этим ничего нельзя поделать. Однако незапрограммированная часть ROM также возвращает $\#FF_H$, и это можно использовать. Предлагаемая технология состоит в том, чтобы записать во все неиспользованные ячейки памяти однобайтовую команду микропроцессора NOP (холостая операция) (рис. 2). В последние несколько ячеек памяти в ROM может быть занесена команда JMP RESET, обычно трехбайтовая, которая будет сбрасывать микропроцессор. Тогда, если микропроцессор по ошибке обращается к неиспользуемой части памяти, он попадает на строку команд NOP и выполняет их (что вполне безопасно), пока не достигнет команды JMP RESET, по которой и перезагрузится. Как один из вариантов, можно предложить заполнение неиспользуемых зон памяти командой STOP вместо NOP, по которой будет срабатывать аппаратный сторож, перезагружающий систему.

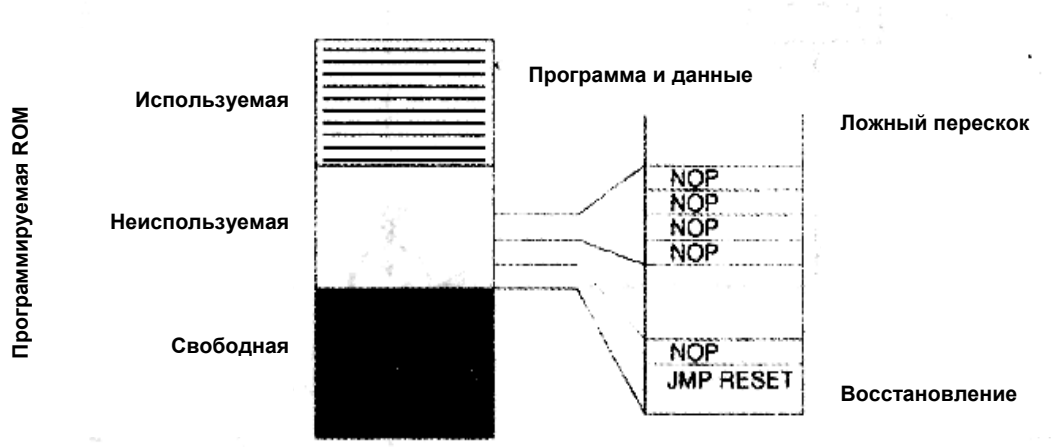

Рис. 2. Защита неиспользуемой памяти с помощью процедуры NOP

Эффективность этой методики зависит от того, какой объем из полного возможного пространства памяти будет заполнен командами NOP или STOP, так как микропроцессор при сбое может обратиться по чисто случайному адресу (на самом деле, ошибки далеко не совсем случайны и, возможно, обусловлены особенно восприимчивым состоянием или конкретным слабым местом линии данных / адреса). Если микропроцессор обращается к несуществующему адресу, результат обращения будет зависеть от того, как он воспримет сообщение $\#FF_n$ и как его обработает. Относительная дешевизна больших ROM и EPROM означает, что вы можете рассматривать их использование, и для формирования файла полной карты памяти с ROM, даже если ваши требования к объему программы незначительны.

6. Повторная инициализация. Так же, как для защиты данных в оперативной памяти, нужно принять меры против искажения установочных состояний программируемых приборов типа портов ввода-вывода или универсальных асинхронных приемопередатчиков.

При этом ошибочно считается, что как только внутренние контрольные регулировки устройства будут установлены (обычно в подпрограмме ини-

циализации), то они так навсегда и сохранят это состояние. Однако под действием помех регистры команд могут изменять свое состояние даже притом, что они не связаны напрямую с внешней шиной. Это может иметь последствия, которые не очевидны для микропроцессора: например, если выход перепрограммируется как ввод, микропроцессор продолжит записывать данные, не зная о своей ошибке.

Самое безопасное решение состоит в том, чтобы периодически реинициализировать все критические регистры, возможно в главной подпрограмме, во время перерывов в ее работе, если они существуют. Таймеры, однако, не могут быть защищены таким образом. Период между поочередными реинициализациями зависит от того, как долго программное обеспечение может допустить существование регистра с испорченным состоянием, с учетом программных накладных расходов, связанных с реинициализацией.

Выводы

Таким образом, были проанализированы основные методы, применяемые для создания программного обеспечения, защищенного от воздействия электромагнитных помех. Выполнена сравнительная характеристика этих методов и определены достоинства и недостатки каждого из них. Рассмотрена возможность применения этих методов в зависимости от типа конкретной помехи.

Литература

1. Рикетс Л.У., Бриджес Дж. Э., Майлетта Дж. Электромагнитный импульс и методы защиты: Пер. с англ. / Под ред Н.А. Ухина. – М.: Атомиздат, 1979. – 328 с.

2. Miller D.A., Bridges J.E. // IEEE Trans. Electromagnetic Compatibility. – 1968. – V. 10, № 1. – P. 52.

3. Европейский стандарт CENELEC EN50173 Performance Requirements of Generic Cabling Schemes «Информационные технологии. Структурированные кабельные системы». – М.: Компьютер Press, 1998. – 384 с.

4. Кудрин И., Найденко В., Прохоров М. Уменьшение информативного побочного электромагнитного излучения при передаче информации последовательным способом // Правове, нормативне та метрологічне забезпечення систем захисту інформації в Україні. – Науково-технічна збірка. – К., 2001. – Вип. № 2. –С. 111-115.

5. Международный стандарт ISO/IES 11801:2002 Edition 2 Information Technology – Generic cabling for customer premises cabling «Информационные технологии. Структурированная кабельная система для помещений заказчиков» – М.: Компьютер Press, 1998. – 384 с.

6. Мырова Т.О., Чепиженко А.З. Обеспечение стойкости аппаратуры связи к ионизирующим и электромагнитным излучениям. – М.: Радио и связь, 1988. – 296 с.

7. Серков А.А. Проблемы обеспечения живучести информационных систем в условиях электромагнитных влияний // Вестник НТУ «ХПИ». Сб. научн. трудов. Тем. вып. Электроэнергетика и преобразующая техника. – Х.: НТУ «ХПИ», 2002. – №7, т.1. – 168 с.

Поступила в редакцию 27.02.2007

Рецензент: д-р техн. наук, проф. В.Д. Дмитриенко, Национальный технический университет «Харьковский политехнический институт», Харьков.