РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ В КРИТИЧЕСКИ ВАЖНЫХ СИСТЕМАХ

Королюк Е.С., Бразовский К.С.
Научный руководитель: Бразовский К.С., к.м.н., доцент
Национальный исследовательский Томский политехнический университет
Россия, г. Томск, пр. Ленина, 30, 634050
E-mail: esk13@tpu.ru

ANALYZING EMBEDDED DEVICE PROGRAMMING SPECIFICATIONS IN CRITICAL SYSTEM TO DESIGN PROGRAMMING GUIDELINES

Korolyuk E.S., Brazovsky K.S.
Scientific Supervisor: Assiciate Professor, Ph.D. Brazovsky K.S.
Tomsk Polytechnic University
Russia, Tomsk, Lenin str., 30, 634050
E-mail: esk13@tpu.ru

Безопасность программного обеспечения является важной характеристикой, показывающей вероятность правильности работы программы при различных условиях эксплуатации. Однако разработчики встроенного программного обеспечения не всегда уделяют должное внимание этому аспекту, особенно в критически важных системах. В данной работе рассматриваются международные стандарты и сертификаты безопасности программного обеспечения. На основе исходного кода снятого с производства электрохирургического медицинского аппарата ЭХВЧ-80 компании НПО «НИКОР» анализируются вопросы обоснованности выбора языка программирования и использованной архитектуры. Приводится алгоритм работы программы, а также описаны основные задачи, выполняемые микроконтроллером. В заключительной части статьи, на основе анализа программного обеспечения автор предлагает рекомендации по упрощению поддержки и возможные варианты уменьшения ошибок, как в коде, так и в архитектуре разрабатываемого приложения. Результаты исследования позволяют повысить безопасность ПО, особенно в критически важных системах.

Software safety is an important characteristic, which indicates the probability of correct program operation under different operating conditions. However, embedded software developers do not always focus on this aspect, especially in critical systems. The present article discusses the international software security standards and certificates. The author studies the source code of the presently discontinued electrosurgical device EHVCh-80 by NPO "NIKOR" and analyzes the issues of validity of choosing the programming language and architecture. The paper provides the algorithm of the program's operation and describes the main tasks performed by the microcontroller. In conclusion, the author uses the results of this software's analysis to offer recommendations to simplify software support and possible options to reduce errors, both in the code and in the architecture of the developed application. The results of this research allow increasing software safety, especially in critical systems.

По мере усовершенствования систем управления во встраиваемых системах, необходимо обеспечивать безопасность и надежность проектируемого программного обеспечения, особенно в системах, критичных к безопасности. При создании медицинского оборудования следует понимать, что опасность для пациента может представлять не только отказ каких-либо изделий (электрических, механических и др.), но и ошибки в его программном обеспечении.

Существует ряд международных стандартов и сертификатов безопасности, например, MISRAC [1] или Safety Integrity Level [2]. При использовании данных стандартов анализ безопасности заключается в уменьшении потенциального источника вреда и сведении его к минимуму, путем рассмотрения этапов проектирования устройства, начиная от разработки и заканчивая выводом из эксплуатации.

Сертификация Safety Integrity Level (SIL) – системный уровень надежности появилась из-за крупнейших аварий, произошедших в промышленности: Бхопальской катастрофы (1984 год), утечки газа на нефтедобывающей платформе Piper Alpha (1988 год) и других. В ходе расследования этих событий были выдвинуты требования со стороны правительственных структур для проверки систем, отвечающих за безопасность. Появилось несколько международных стандартов ISA-S84 [3] и IEC 61508/61511[4], вошедших в основу отечественных стандартов, например, ГОСТ Р МЭК 61508-1-2007 [5]. Значение уровня надежности зависит от того, будет ли система подвергаться менее или более строгим требованиям.

Стандарт MISRA является набором дополнительных правил (требований) для разработки программного обеспечения на языке СИ в дополнение к стандартным правилам.

Использование языка высокого уровня, при написании программ для микроконтроллера добавляет некоторые преимущества по сравнению с языком ассемблера. Языка ассемблера имеет свои достоинства. Главными являются быстрота выполнения и малый размер кода, однако и недостатки в виде плохой читаемости и трудности поддержки, что добавляет дополнительных трудностей, если необходимо добавление новых функций или изменения структуры программы. Неправильно используемая команда может изменить данные и вызвать неправильную работу прибора. Из-за жесткой привязанности языка к определенной платформе, невозможно производить автоматический анализ кода на безопасность.

Причины популярности языка СИ во встраиваемых системах просты. Конструкции языка легко сопоставляются машинным инструкциям, язык стандартизирован. Программный код становится независим от типа контроллера и возможен перенос программы на разные платформы.

Помимо ошибок в синтаксисе существуют другие проблемы, где стандарт ISOC [6] указывает, что реализация может быть определенной или неопределенной. Начинающие программисты сталкиваются с трудностями применения этого языка программирования. Решением этого является ограничение использования некоторого функционала языка с четко описанными возможностями повышения производительности приложений. Для решения данных проблем Ассоциация по разработке надежного программного обеспечения для автомобильной промышленности (MISRA) выпустила ряд обязательных, требуемых и рекомендованных правил программирования [1].

Медицинская промышленность производит продукцию, которая напрямую воздействует на пациента. Строгие правила обеспечения безопасности, необходимые во время выполнения процедур, требуют дополнительного контроля в течение всего срока жизни изделия, в том числе к программному обеспечению. Жизненный цикл изделия можно разделить на следующие этапы:

- Научно исследовательская работа
- Опытно конструкторская разработка
- Серийное производство
- Поставка
- Эксплуатация
- Ремонт
- Утилизация

В результате, из-за дополнительных требований по внедрению между этапами жизненного цикла проходят длительные промежутки времени. Это обусловлено не только длительностью сертификации, но и необходимость получения прибыли разработчиком для последующей разработки. Внедрение новых функций и усовершенствование оборудования также является необходимостью. Во время опытно-конструкторской разработки, взятая с запасом, система управления (обычно микроконтроллер или микропроцессов) к моменту необходимого усовершенствования устаревает и для внедрения новых функций программисту приходится использовать дополнительные приемы и нестандартные методы в программировании. Для анализа требований, которые необходимо выдвигать в начале этапа разработки, рассмотрим снятый с производства электрохирургический медицинский аппарат ЭХВЧ-80 компании НПО «НИКОР».

Главным элементом системы управления является микроконтроллер ATmega8 [7]. Программа написана на языке ассемблера в среде AVRStudio. Микроконтроллер выполняет следующие задачи:

• АЦП снимает параметры с датчиков, прикрепленных к пациенту. При завершении преобразования происходит прерывание.

• С помощью ШИМ происходит регулировка мощности выходного сигнала

• При переполнении таймера 0 вызывается прерывание и происходит опрос клавиатуры.

• При переполнении таймера 1 вызывается прерывание и происходит опрос данных с пациента, если прибор не используется, то отключается силовая часть.

• Таймер 2 задействован для сервисных целей. При его переполнении вызывается прерывание и происходит чтение/запись данных для настройки.

• Внешние прерывания срабатывают при нажатии кнопок управления непосредственно на инструменте.

• Выводы общего назначения используются для управления индикаторами, клавиатурой и элементами на силовой части прибора.

Программа работает по следующему алгоритму:

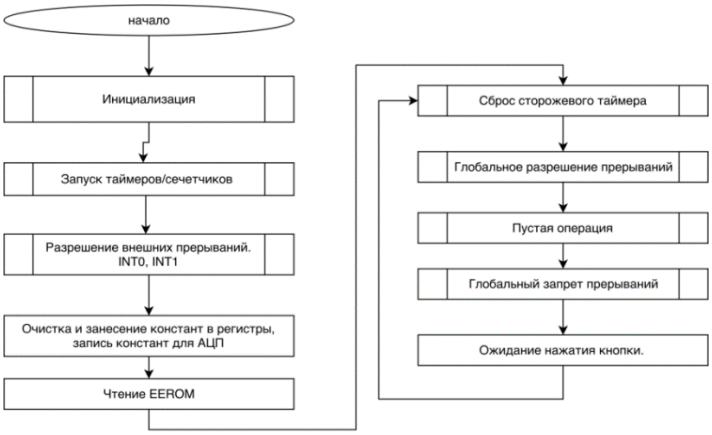

Рис. 1. Алгоритм работы программы аппарата ЭХВЧ-80

Основные действия совершаются во время глобального разрешения прерываний. В это время выполняются условия срабатывания прерываний, в которых выполняются все основные процедуры, и задействуются регистры для флагов, с помощью которых в других участках кода проверяются режимы работы устройства. Реализуется это следующим образом:

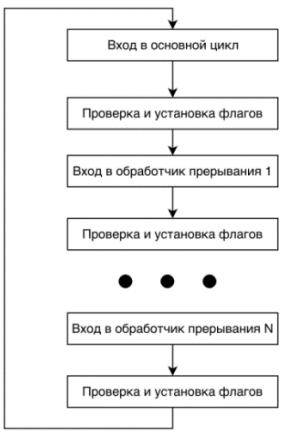

Рис. 2. Алгоритм работы прерываний аппарата ЭХВЧ-80

Используемая структура удобна, когда размер программного кода небольшой. При увеличении его размеров появляются проблемы с отслеживанием флагов. Ошибка с флагом может привести к установлению неправильного режима работы прибора. Микроконтроллер Atmega8 имеет 31 доступный регистр, из них программист использовал 4 регистра, что соответствует 32 флагам. При изменении структуры программы и дополнительной поддержки программы необходимо знать назначение каждого флага.

Заключение

В результате анализа архитектуры и программного кода медицинского аппарата ЭХВЧ-80,было выявлено, что в целях повышения надежности написания программного обеспечения во встроенных системах можно сделать следующие рекомендации:

1. Не стоит возлагать большую уверенность на программное обеспечение. Разработчики не должны быть слишком уверены в корректности программного обеспечения. Для облегчения понимания кода не нужно его обуфицировать (запутывать).

2. Необходимость защитного проектирования. Программное обеспечение должно содержать процедуры самопроверки. Необходимо вести журналы работы.

3. Предупредить причины опасных отказов. Хоть и в описанном аппарате предусмотрены независимые предохранители, отключающие аппарат, разработчикам важно помнить об их наличии.

4. Не применять плохие практики программирования. Необходимо, чтобы назначение каждой переменной, управляемой устройством было понятно, и функции, возложенные на нее, описаны. Никогда не стоит использовать одну переменную для управления устройством, особенно это важно в критически важных системах.

СПИСОК ЛИТЕРАТУРЫ

1. MIRA Limited. "MISRA-C: 2004 Guidelines for the use of the C language in critical systems." Edition 2. Warwickshire, UK: MIRA Limited, July 2008 (ISBN 978-0-9524156-4-0).

2. Manual Safety Integrity Level. Andy Ingrey, Patrick Lerévérend, Dr. Andreas Hildebrandt [электронный ресурс] – Режим Доступа http://goo.gl/IrxoKt . Дата доступа: 10.03.16

3. ANSI/ISA 84.00.01-2004 and Existing Safety Instrumented Systems.

4. IEC 61508/61511 SafetyIntegrityLevel [электронный ресурс] - Режим Доступа http://goo.gl/lWVZOA . Дата доступа: 10.03.16.

5. ГОСТРМЭК 61508-2-2012 Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 2. Требования к системам. – М. :Стандартинформ, 2014

6. ISO/IEC 9899:2011 — Information technology — Programming Languages — C.

7. ATmega8535-16PU Atmel 8-bit Microcontrollers [электронный ресурс] –Режим Доступа http://www.atmel.com/images/doc2502.pdf . Дата доступа: 10.03.16.