# Обновление программного обеспечения

## распределенных микропроцессорных систем

**Современные условия вынуждают коллективы разработчиков микропроцессорных систем резко сокращать длительность этапа проектирования и в кратчайшие сроки выходить на этап пуско-наладки и эксплуатации. При этом разработчику приходится жертвовать функциональностью и качеством создаваемого программного обеспечения, чтобы уложиться в заданные сроки. Одним из способов исправить ситуацию является предоставление возможности обновления программного обеспечения микропроцессорных модулей системы после того, как она была сдана в эксплуатацию и установлена на объекте.**

**Николай Постников**

pnp@d1.ifmo.ru

### Механизмы внутрисхемного программирования

В общем случае проблема обновления программного обеспечения микропроцессорного модуля сводится к замене содержимого (обычно это объектный код программы микроконтроллера) энергонезависимой памяти самого микроконтроллера или внешних микросхем. В настоящее время технологии FLASH, EEPROM, FRAM позволяют без демонтажа и физической замены микросхемы программировать носитель информации средствами самого микроконтроллера, не привлекая дополнительного оборудования или инструментария. Ведущие производители микропроцессорной элементной базы предлагают различные способы перепрограммирования своих устройств, уже установленных на плату [1]. Такой способ программирования называется In-Circuit Programming (ICP) или In-System Programming (ISP) и обычно используется для первоначальной инициализации микроконтроллера. Если разработчик использует микроконтроллерную базу, поддерживающую тот или иной способ ISP, и в процессе эксплуатации системы имеется доступ к инструментальному ISP-каналу каждого микроконтроллера, то в этом случае проблема обновления программного обеспечения может быть решена «стандартными» средствами (средствами, предоставленными производителями микропроцессоров), хотя это и сопряжено с определенными неудобствами:

- для обеспечения работы ISP-канала может потребоваться дополнительное оборудование и инструментальные средства;
- перевод микроконтроллера в режим ISP зачастую ставит сложные схемотехнические задачи перед разработчиком, что повышает стоимость устройства;
- различные узлы системы могут иметь свой уникальный ISP-интерфейс и инструментарий.

Даже будучи относительно неудобным, описанный способ позволит решить проблему обновления программного обеспечения компактных одномодульных систем. Для более сложных распределенных микропроцессорных систем такой способ становится очень ресурсоемким и сложным, как для разработчика, так и для конечного пользователя.

### Распределенные микропроцессорные системы

Из множества микропроцессорных систем можно выделить достаточно широкий класс распределенных информационно-управляющих систем. Основной задачей информационно-управляющей системы является получение информации о состоянии внешнего объекта посредством специальных датчиков и сенсоров, принятие решения на основе полученной и накопленной информации и воздействие на объект с целью перевода его в заданное состояние. В некоторых случаях объект управления может простираться в пространстве на значительные расстояния или же ввиду жестких условий эксплуатации, ограничений габаритов, климатических требований, нехватки вычислительных ресурсов невозможно сосредоточить всю систему управления на одном-единственном вычислителе. В этом случае разработчик вынужден иметь дело со сложным иерархическим комплексом вычислительных

узлов, объединенных некоторой информационной сетью и решающих общую задачу.

Для распределенных микропроцессорных систем базовые механизмы внутрисхемного программирования практически неприменимы по следующим причинам:

- разработчик не имеет физического доступа к отдельным модулям системы или такой доступ сопряжен со значительными финансовыми и иными затратами;
- ISP-канал вычислителей был использован при начальном программировании микроконтроллеров и в целях повышения безопасности был отключен при сдаче системы в эксплуатацию;
- в состав системы может входить огромное количество модулей, индивидуальное программирование которых не представляется возможным (перепрограммированию подлежат как функционирующие в конкретный момент времени контроллеры, так и контроллеры, находящиеся на сервисном обслуживании или в ЗИП);
- выбор удобного времени перепрограммирования того или иного модуля (временно его функции будут недоступны системе) в общем случае целиком зависит от самой системы.

Перечисленные проблемы заставляют разработчика рассмотреть задачу обновления программного обеспечения микроконтроллеров не с точки зрения отдельных модулей, а с точки зрения системы. Задачу можно сформулировать как обновление (замену) программного обеспечения отдельных модулей системы, не нарушая общей работоспособности системы в целом. При решении этой задачи разработчику необходимо учитывать определенные требования:

- *Удобство использования.* Для проведения обновления системного программного обеспечения конечный пользователь должен прилагать минимум усилий и совершать минимум действий.
- *Единый инструментальный интерфейс.* Несмотря на возможную широкую номенклатуру устройств в системе, интерфейс обновления системного программного обеспечения должен быть единым для всех вычислителей.
- *Малая ресурсоемкость.* Это требование очень важно, т. к. возможность обновления программного обеспечения в общем случае не является необходимой частью целевой системы, а является способом ускорения проведения цикла разработки и выдачи изделия на рынок (конечному пользователю). В таких условиях отвлекать на решение этой задачи значительные вычислительные ресурсы нерационально.
- *Безопасность.* Обеспечив микроконтроллер в составе системы возможностью самостоятельно изменять программное обеспечение, разработчик сталкивается с проблемой возможных сбоев, которые приведут к несанкционированному выполнению тех или иных процедур обновления программного обеспечения, что в свою очередь приведет к нарушению целостности программного кода микроконтроллера.

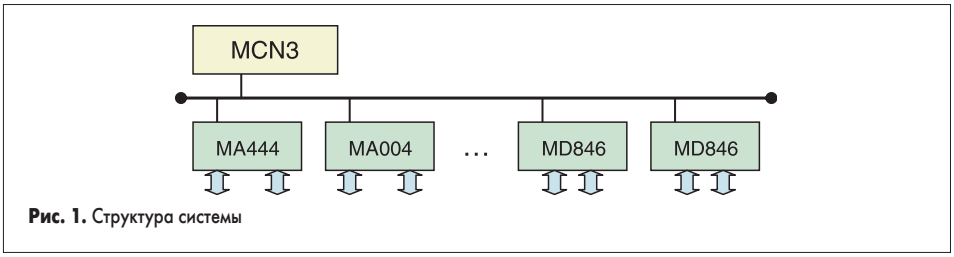


**Рис. 1.** Структура системы

- *Надежность.* Даже в случае корректной работы процедур внутреннего перепрограммирования энергонезависимых ресурсов в принципе не должно складываться ситуации, когда в результате того или иного сбоя (потеря питания, ошибка связи, рестарт микроконтроллера) микропроцессорный модуль останется без программного обеспечения и тем самым будет выведен из строя.

Для отдельных микроконтроллеров производители (Philips Semiconductors, Microchip, Atmel, Motorola, ST и др.) предлагают решение обозначенной задачи средствами In Application Programming (IAP). К таким средствам относятся различные аппаратные решения, предусмотренные производителями микроконтроллеров для перепрограммирования ресурсов энергонезависимой памяти прямо из целевой задачи без перевода микроконтроллера в специальный режим работы.

Таким образом, задачу обновления программного обеспечения распределенных микропроцессорных систем разработчик решает на двух уровнях: на системном уровне и на уровне микропроцессорного узла системы.

На первом уровне необходимо обеспечить взаимодействие частей системы в рамках единого инструментального протокола, идентификацию типов узлов и версий системного программного обеспечения, доставку новых версий программного обеспечения до вычислительных узлов. При этом общая работоспособность системы не должна быть нарушена. На этом уровне важными являются требования к удобству использования и единому инструментальному интерфейсу.

На уровне вычислительного узла необходимо обеспечить защиту программного кода, проверку целостности доставленного программного обеспечения, перепрограммирование энергонезависимых ресурсов микроконтроллера. На этом уровне важными являются требования к малой ресурсоемкости, безопасности и надежности.

## Пример реализации технологии обновления программного обеспечения

Одна из технологий обновления программного обеспечения распределенных микропроцессорных систем была реализована специалистами ООО «ЛМТ» (http://lmt.ifmo.ru) в комплексе технических средств КТЖ-2 (КТС КТЖ-2). В состав комплекса входят модули центрального вычислителя MCNx, периферийные модули MAxxx, MDxxx и CSCx, коммуникационная система на базе интерфейсов Ethernet, CAN и RS-232/485, инструментальные средства системного и прикладного программирования (поддерживающие стандарт IEC 1131-3). Центральный вычислитель MCN3 на базе 32-разрядного микроконтроллера Fujitsu MB91F362, оснащенный графическим дисплеем и сенсорной панелью, может выполнять функции ПЛК, интеллектуального пульта оператора, сетевого шлюза. При работе MCN3 взаимодействует с множеством периферийных модулей (MA444, MD846, MA004 и др.), обеспечивающих дискретный и аналоговый ввод-вывод, замер физических параметров окружающей среды и т. д. Конструктивно центральные модули выполнены в виде компактных панельных компьютеров, периферийные модули оформлены в корпусах для монтажа на DIN-рейку. Типы и количество периферийных модулей могут с легкостью варьироваться и расширяться. Типовая структура системы на базе КТС КТЖ-2 представлена на рис. 1.

Большинство типов периферийных модулей системы построено на базе 16-разрядных микроконтроллеров Fujitsu MB90F598. Семейство микроконтроллеров MB90595 предлагает механизмы защиты энергонезависимой памяти, но для повышения уровня безопасности функционирования периферийных модулей процедуры, которые работают с внутренней Flash-памятью микроконтроллера, были исключены из состава целевой системы. Код процедур становится доступным только после санкционированного и корректного получения программного обеспечения периферийным модулем.

Весь процесс обновления программного обеспечения был разбит на независимые этапы. При этом процесс обновления программного обеспечения может начинаться с любого из этапов, а сбой на любом из этапов фатально не нарушит весь процесс обновления, а лишь потребует повторного прохождения сбойного этапа. Реализовано 4 этапа:

1. Создание образа системного программного обеспечения (обновления). Эти данные помимо самого объектного кода содержат команды для его размещения во Flash-памяти периферийного модуля, различные контрольные суммы, идентификаторы типов модулей и версии программного обеспечения. Обновление формируется таким образом, что содержащихся в нем данных полностью хватает для восстановления объектного кода программного обеспечения периферийного модуля.
2. Доставка обновления в систему по специальному инструментальному каналу. Передаваемое обновление может воспринимать как центральный вычислитель, так и периферийные модули.

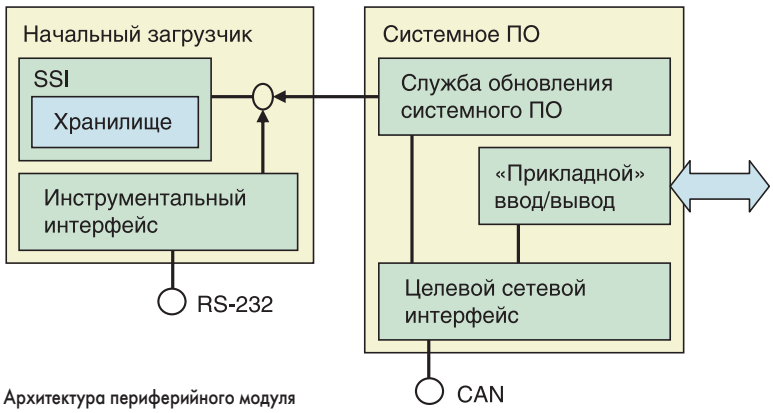


**Рис. 2.** Архитектура периферийного модуля

3. Передача обновления в периферийные модули системы, которые сохраняют его для последующей обработки.
4. В каждом модуле происходит обработка обновления, полное или частичное уничтожение старого программного обеспечения модуля и программирование нового программного обеспечения.

Поскольку обновление должно быть полностью доставлено и сохранено перед началом его обработки в периферийном модуле, пришлось учесть требования малой ресурсоемкости. Технически задача была решена сокращением объема обновления с использованием относительно несложных алгоритмов сжатия данных без потерь. На персональном компьютере разработчика создается файл-обновление, который потом нужно без изменений доставить в периферийный модуль. Файл-обновление самодостаточен, что позволяет периферийному модулю в дальнейшем самостоятельно восстановить объектный код программного обеспечения.

Все программное обеспечение периферийного модуля разбито на две части: начальный загрузчик и системное программное обеспечение (собственно эту часть программного обеспечения имеет возможность обновлять разработчик или пользователь). Архитектура периферийного модуля представлена на рис. 2. Передача и сохранение обновления реализуется через специально созданный программный интерфейс работы с хранилищем обновлений (SSI). Этот интерфейс содержит всего 5 функций (SsiUnlock, SsiAllocate, SsiStoreItem, SsiGetItemCount, SsiLock), но при этом полностью обеспечивает решение поставленной задачи. Интерфейс SSI экспортируется через собственный инструментальный канал узла (реализуется в начальном загрузчике) и через целевой протокол сети CAN (реализуется в системном программном обеспечении).

Технические характеристики периферийных модулей, относящиеся к работе механизмов обновления программного обеспечения, приведены в таблице.

**Таблица.** Технические характеристики периферийного модуля

| | |
|---|---|
| Объем кода обновляемого программного обеспечения | 56 кбайт |
| Объем кода начального загрузчика | 4 кбайт |
| Объем резидентной части загрузчика в RAM | 200 байт |
| Объем хранилища | 64 кбайт |
| Типичный объем обновления | 10–20 кбайт |
| Задержка при рестарте модуля<br>• с декодированием обновления<br>• без декодирования обновления | 15–20 с<br>нет |

В целевом протоколе сети существует возможность определения номенклатуры периферийных модулей системы в реальном масштабе времени, определение их типов и версий программного обеспечения. Передача обновлений периферийным модулям происходит в широковещательном режиме, что очень важно, так как в системе обычно присутствует много однотипных модулей. Передача обновлений по сети осуществляется в наименее приоритетном режиме, что практически не влияет на временные параметры выполнения системой целевой задачи. Служба, реализованная в центральном вычислителе, позволяет в реальном масштабе времени в полуавтоматическом режиме отслеживать версии программного обеспечения периферийных модулей и производить при необходимости процедуру загрузки обновлений.

## Заключение

Использование технологий обновления программного обеспечения в распределенных микропроцессорных системах позволяет разработчикам значительно сократить сроки получения пилотного образца системы, снизить затраты на этапах внедрения и опытной эксплуатации, получать масштабируемые решения с гибкой целевой функциональностью [2]. Разработанная технология обновления программного обеспечения может быть легко реализована в различных коммуникационных средах.

## Литература

1. Гаврилюк Д. Внутрисхемное программирование // Компоненты и Технологии. 2003. № 8.
2. Платунов А. Е., Постников Н. П. Единое проектное пространство плюс аспектная технология — перспективная парадигма проектирования встраиваемых систем // Научно-технический вестник СПбГУ ИТМО. Вып. 11. Актуальные проблемы анализа и синтеза сложных технических систем. СПб.: СПбГУ ИТМО. 2003.