Способ эмуляция EEPROM во Flash-памяти микроконтроллера

Иосиф Каршенбойм, iosifk@eltech.spb.ru

Часто при работе пользовательской программы возникает необходимость хранить небольшой объем данных и периодически обновлять их в процессе работы устройства. К таким данным обычно относятся параметры калибровки узлов устройства, содержащего данный микроконтроллер. Под словом “хранить” здесь понимается энергонезависимое хранение. Обычно для этой цели используется EEPROM. Однако, в том случае, когда в микроконтроллере нет встроенного блока EEPROM или надежность его работы недостаточна, то прибегают к режиму эмуляции EEPROM во Flash-памяти микроконтроллера.

Чем же отличаются EEPROM и Flash? В табл. 1 приведены сравнительные характеристики для этих двух типов памяти.

Таблица 1 **Сравнительные характеристики для памяти EEPROM и Flash**

Тип памяти	Производитель; Наименование микросхемы	Способ доступа: чтение / запись	Минимальный объём стираемой информации	Число циклов перезаписи	Гарантированная длительность хранения (температура)	Рабочий диапазон температур
EEPROM	Atmel	байтами	1 Байт	100000	–	–
Flash	Atmel; AT90CAN128 Automotive	байтами	Блок 256 Байт	10000	–	–
	NEC; 78K0S/KY1	байтами	Блок 256 Байт	1000	10 лет (85°C)	–40°…+105°C

Примечание: для сравнения взяты микросхемы, предназначенные для применения на автотранспорте, см. Л[1–3].

Как видно из табл. 1, память типа EEPROM и Flash имеют два основных отличия. Первое – это число циклов перезаписи. У EEPROM это число на один-два порядка больше, чем у памяти Flash. Второе, и наиболее существенное, – память EEPROM позволяет многократно перезаписывать данные, находящиеся по одному и тому же адресу без каких либо дополнительных процедур, а для памяти типа Flash перед каждой последующей записью данных в одну и ту же ячейку необходимо произвести стирание блока памяти, в котором находится данная ячейка памяти.

Данная статья является частью описания эмуляции EEPROM, приведенной в Л[3] . В Л[3] рассмотрен пример эмуляции EEPROM для двух вариантов форматов данных пользователя:

- первый вариант – необходимо записывать данные фиксированной длины;
- второй пример – необходимо записывать данные переменной длины.

В данной статье рассмотрен только первый случай – запись данных фиксированной длины – 2 байта.

Поле памяти микроконтроллера представляет собой набор блоков по 256 байт. Как было описано выше, данные могут записываться в различные ячейки памяти последовательно цикл за циклом, а стираться данные могут только одновременно блоками по 256 байт.

В одну ячейку можно осуществить только одну запись, далее эту ячейку необходимо стереть. После этого в эту ячейку можно осуществлять следующую запись. Следовательно, невозможно организовать один указатель на последнюю запись, поскольку его пришлось бы переписывать в каждом цикле записи.

Поэтому данные предлагается записывать кадрами, состоящими из следующих полей: поле номера кадра, поля данных и поля разделителя. И вся информация о кадре данных должна храниться в самом кадре.

Предлагается простой алгоритм кодирования полей кадров, позволяющий определить последнюю правильно выполненную запись. Таким образом, если необходимо записать 2 байта данных, то для записи одного кадра потребуется, с учетом служебной информации в кадре, 4 байта.

Кроме того, для индикации состояния блока необходимо выделить еще 2 байта. Таким образом, в одном блоке памяти из 256 байт и при необходимости хранения 2 байт данных в каждом кадре данных, можно разместить 63 кадра данных.

Для правильного хранения данных при выполнении операции стирания в блоке, требуется, как минимум, два таких блока памяти. Блоки памяти, используемые для эмуляции EEPROM, можно расположить в любой свободной области Flash-памяти. На рис. 1 показано поле памяти микроконтроллера 78K0S/K, а именно пользовательская программа – 3,5 Кбайт и блоки 14 и 15, использующиеся для эмуляции EEPROM.

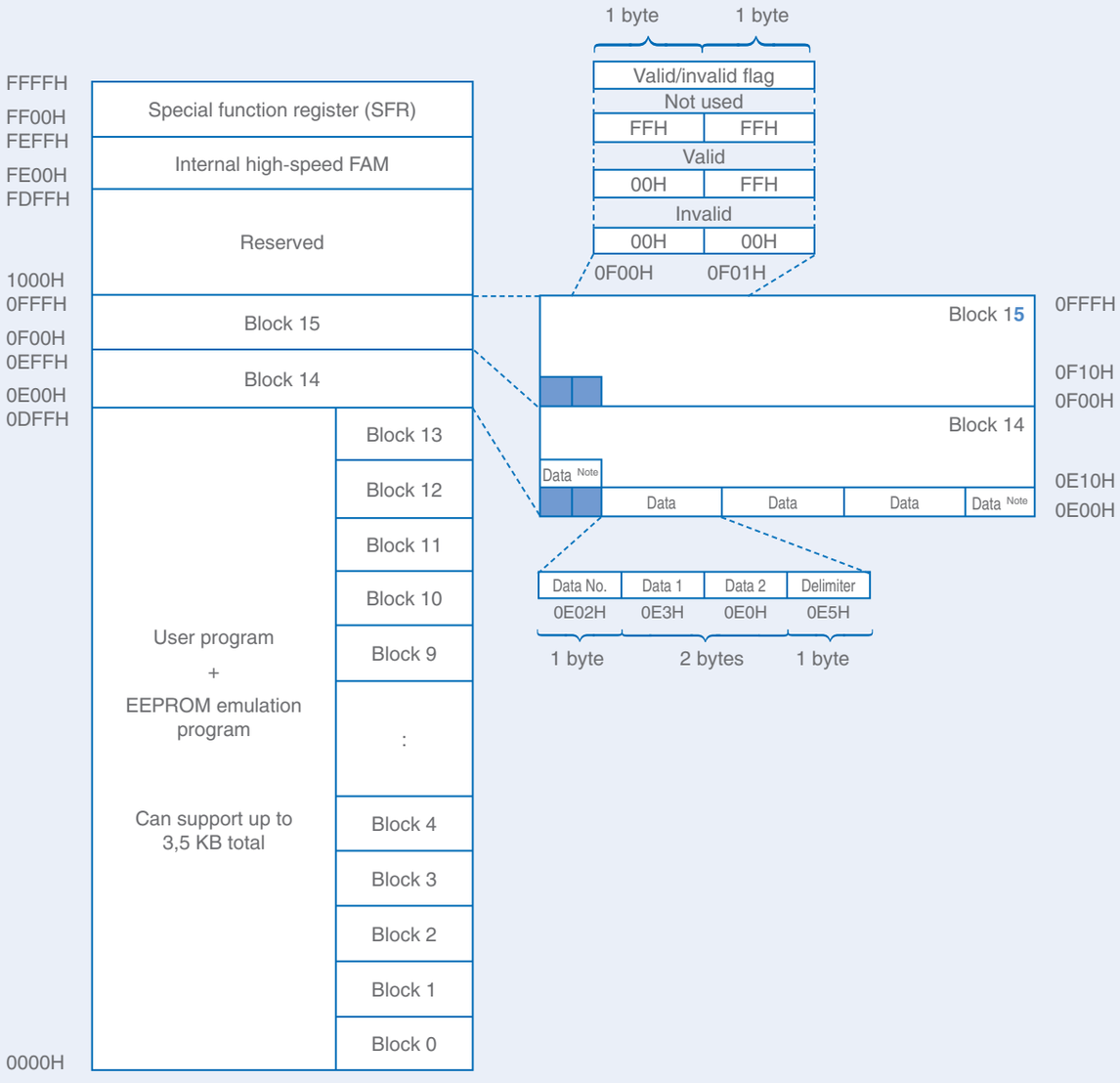

Рис. 1. Поле памяти микроконтроллера 78K0S/K. Блоки 14 и 15 выделены для использования в режиме эмуляции EEPROM

СТРУКТУРА КАДРА ДАННЫХ

Как было сказано выше, данные будут записываться кадрами, каждый кадр будет состоять из четырех байтов: номер кадра данных (1 байт), данные (2 байта), разделитель (1 байт).

НОМЕР КАДРА ДАННЫХ

Номер кадра данных используют для того, чтобы отличить один кадр данных от другого при записи или чтении. Поскольку, после того, как флэш-память стерта, все данные, находящиеся во флэш-памяти имеют значение FFH, и можно использовать любое значение номера кадров данных от 00 до FEH.

Перед очередной записью данных, пользователь должен назначить номер кадра на каждый новый кадр данных. Таким образом, каждый раз, когда программа записи будет выполнена, данные будут записаны в 4-байтовых кадрах, начинающихся с начала блока.

Чтобы считать последние записанные данные, надо произвести поиск последнего записанного кадра. Последний записанный кадр ищется от начала блока и до последнего записанного кадра. Если найдены несколько кадров данных, имеющих одинаковый номер кадра, то достоверными будут считаться те данные, который расположены ближе к тому месту в памяти, где записаны последние данные.

Если номер данных читается как FFH, то это воспринимается как место в памяти, где ничего не было записано. Этот адрес принимается как пункт конца записи данных, и поиск считается завершенным. Исходя из этого, только кадры данных, имеющие номера от 00 до FEH будут достоверны.

РАЗДЕЛИТЕЛЬ

Разделители используются для того, чтобы иметь возможность обнаружить неудачную запись данных, в тех случаях, когда в течение операции записи данных произошло нарушение электропитания и/или имели место другие проблемы, которые могли привести к аналогичному результату. Правильное окончание цикла записи данных, обычно определяется по тому, насколько правильно произведена запись разделителя, поскольку в эту область данные записываются последними. Значение разделителя должно читаться как 00.

Если разделитель не может прочитаться правильно, т. е. его код не 00, то, вероятно, что произошла авария при записи данных, поэтому соответствующие данные не используются.

Если поиск находит неправильно записанный разделитель в последних кадрах данных, то данные, записанные в предыдущем цикле записи, считаются как последние достоверно записанные данные.

Флаги достоверности и недостоверности для блока памяти

Для того, чтобы зафиксировать текущее состояние блока данных, используются два флага: флаг достоверности (Valid flag – ФД) и флаг недостоверности (Invalid flag – ФН).

Блоки памяти, используемые для эмуляции EEPROM, имеют байтовую структуру, поэтому и флаги – ФД и ФН имеют так же байтную структуру. Эти два флага помещаются в начале блока памяти. Состояние флагов описывает состояние соответствующих блоков памяти так, как указано в табл. 2.

Таблица 2 Состояние флагов достоверности для различных состояний блоков памяти

Флаги	ФД	ФН	Состояние блока памяти
Данные, читаемые из памяти	FFH	FFH	Исходное состояние после стирания данных из блока
	00	FFH	Данные, записанные в этом блоке – достоверны
	00	00	Данные, записанные в этом блоке – не достоверны
	FFH	00	Данные, записанные в этом блоке – не достоверны

АЛГОРИТМ ХРАНЕНИЯ ДАННЫХ И ОПЕРАЦИЙ ПОИСКА, КОТОРЫЙ ДОЛЖЕН ВЫПОЛНЯТЬСЯ ПРИ ОТСУТСТВИИ ОШИБОК

Рассмотрим алгоритм хранения данных и операций поиска, который должен выполняться при отсутствии ошибок. Этот алгоритм будет рассмотрен на примере использования блоков памяти 14 и 15, см. рис. 1, задействованных для эмуляции EEPROM.

Шаг 1. Блок 14 Установлен байт статуса, блок стал разрешенным для записи

Смещение	+0	Абс. адрес
Блок 14	Данные	
ФД	**00H**	0E00H
ФН	FFH	0E01H
Данные “a”	FFH	0E02H

Шаг 2. Блок 14 Записаны данные под номером 1. Данные 11H и 22H

Смещение	+0	+1	+2	+3	Абс. адрес
Блок 14	Данные				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	**01H**	**11H**	**22H**	**00H**	0E02H
Данные “b”	FFH				0E06H

Шаг 3. Блок 14 — **Записаны данные под номером 2. Данные 22H и 33H**

Смещение	+0	+1	+2	+3	Абс. адрес
Блок 14	Данные				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	11H	22H	00H	0E02H
Данные “b”	**02H**	**22H**	**33H**	**00H**	0E06H
Данные “c”	FFH				0E0AH

Шаг 4. Блок 14 — **Записаны данные под номером 2. Данные 20H и 30H**

Смещение	+0	+1	+2	+3	Абс. адрес
Блок 14	Данные				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	11H	22H	00H	0E02H
Данные “b”	02H	22H	33H	00H	0E06H
Данные “c”	**02H**	**20H**	**30H**	**00H**	0E0AH
Данные “d”	FFH				0E0EH

Шаг 5. Блок 14 — **Прочитаны данные под номером 2. Данные 20H и 30H**

Смещение	+0	+1	+2	+3	Абс. адрес
Блок 14	Данные				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	11H	22H	00H	0E02H
Данные “b”	02H	22H	33H	00H	0E06H
Данные “c”	02H	20H	30H	00H	0E0AH
Данные “d”	FFH				0E0EH

В этом шаге производится поиск последнего записанного кадра. При чтении данных производится поиск последнего записанного номера кадра, отличного от FFH. В данном случае им будет кадр с данными “a”, так как код номера кадра для этих данных не равен FFH, что свидетельствует о том, что запись этих данных производилась.

Далее проверяется код разделителя. Для строки “a” код разделителя записан правильно. Поэтому программа переходит к следующей строке. Аналогично проверяется строка “b”. В этой строке все параметры записаны правильно.

Программа переходит к поиску следующей записи. Но в строке “d” запись не производилась, поэтому читается код FFH, и, следовательно, за достоверные данные принимается не строка “d” а предыдущая строка, то есть строка “c”.

АЛГОРИТМ ХРАНЕНИЯ ДАННЫХ И ОПЕРАЦИЙ ПОИСКА, КОТОРЫЙ ДОЛЖЕН ВЫПОЛНЯТЬСЯ ПРИ ОШИБКАХ ЗАПИСИ ДАННЫХ В ПАМЯТЬ

Далее описывается работа алгоритма эмуляции EEPROM в тех случаях, когда имели место аварии записи, вызванные прерыванием электропитания или другими причинами.

Шаг 1. Блок 14 — **Установлен байт статуса, блок стал разрешенным для записи**

Смещение	+0	Абс. адрес
Блок 14	Данные	
ФД	**00H**	0E00H
ФН	FFH	0E01H
Данные	FFH	0E02H

Шаг 3. Блок 14 — **Записаны данные под номером 2. Данные 22H и 33H**

Смещение	+0	+1	+2	+3	Абс. адрес
Блок 14	Данные				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	11H	22H	00H	0E02H
Данные “b”	**02H**	**22H**	**33H**	**00H**	0E06H
Данные “c”	FFH				0E0AH

Шаг 4. Блок 14 — **Записаны данные под номером 2. Данные 20H и 30H**

Смещение	+0	+1	+2	+3	Абс. адрес
Блок 14	Данные				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	11H	22H	00H	0E02H
Данные “b”	02H	22H	33H	00H	0E06H
Данные “c”	**02H**	**20H**	**30H**	**00H**	0E0AH
Данные “d”	FFH				0E0EH

Шаг 5. Блок 14 — **Прочитаны данные под номером 2. Данные 20H и 30H**

Смещение	+0	+1	+2	+3	Абс. адрес
Блок 14	Данные				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	11H	22H	00H	0E02H
Данные “b”	02H	22H	33H	00H	0E06H
Данные “c”	02H	20H	30H	00H	0E0AH
Данные “d”	FFH				0E0EH

В этом шаге производится поиск последнего записанного кадра. При чтении данных производится поиск последнего записанного номера кадра, отличного от FFH. В данном случае им будет кадр с данными “a”, так как код номера кадра для этих данных не равен FFH, что свидетельствует о том, что запись этих данных производилась.

Далее проверяется код разделителя. Для строки “a” код разделителя записан правильно. Поэтому программа переходит к следующей строке. Аналогично проверяется строка “b”. В этой строке все параметры записаны правильно.

Программа переходит к поиску следующей записи. Но в строке “d” запись не производилась, поэтому читается код FFH, и, следовательно, за достоверные данные принимается не строка “d” а предыдущая строка, то есть строка “c”.

АЛГОРИТМ ХРАНЕНИЯ ДАННЫХ И ОПЕРАЦИЙ ПОИСКА, КОТОРЫЙ ДОЛЖЕН ВЫПОЛНЯТЬСЯ ПРИ ОШИБКАХ ЗАПИСИ ДАННЫХ В ПАМЯТЬ

Далее описывается работа алгоритма эмуляции EEPROM в тех случаях, когда имели место аварии записи, вызванные прерыванием электропитания или другими причинами.

Шаг 1. Блок 14 — **Установлен байт статуса, блок стал разрешенным для записи**

Смещение	+0	Абс. адрес
Блок 14	Данные	
ФД	**00H**	0E00H
ФН	FFH	0E01H
Данные	FFH	0E02H

Шаг 2. Блок 14		Записаны данные под номером 1. Данные 11H и 22H			
Смещение	**+0**	**+1**	**+2**	**+3**	**Абс. адрес**
Блок 14	**Данные**				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	**01H**	**11H**	**22H**	**00H**	0E02H
Данные “b”	FFH				0E06H

При чтении данных производится поиск последнего записанного кадра. Потом проверяется разделитель. В этом примере есть разрешенный номер кадра данных и разделитель записан правильно, т. е. его код равен 00. Следовательно, последний кадр данных 01H+11H+22H+00H и операция записи выполнилась успешно.

Шаг 3. Блок 14	Сбой по питанию произошел в то время, когда производилась запись данных под номером 1. В результате этого неправильно записался разделитель. Вместо 00 записалось 01				
Смещение	**+0**	**+1**	**+2**	**+3**	**Абс. адрес**
Блок 14	**Данные**				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	11H	22H	00H	0E02H
Данные “b”	**01H**	**22H**	**33H**	**01H**	0E06H
Данные “c”	FFH				0E0AH

Шаг 4				Читаются данные номер 1	
Смещение	**+0**	**+1**	**+2**	**+3**	**Абс. адрес**
Блок 14	**Данные**				
ФД	00H				0E00H
ФН	FFH				0E01H
Данные “a”	01H	**11H**	22H	**00H**	0E02H
Данные “b”	01H	22H	33H	01H	0E06H
Данные “c”	FFH				0E0AH

В этом шаге производится поиск последнего записанного кадра. При чтении данных производится поиск последнего записанного номера кадра, отличного от FFH. В данном случае им будет кадр с данными “a” “b”, так как код номера кадра для этих данных не равен FFH, что свидетельствует о том, что запись этих данных производилась.

Далее проверяется код разделителя. Для строки “a” код разделителя записан правильно. Поэтому программа переходит к следующей строке. Для строки “b” код номера кадра тоже записан правильно. Но разделитель записан неправильно, т. е. его код не равен 00. Следовательно, этот кадр данных 01H+11H+22H+01H записан неверно.

Программа переходит к поиску следующей записи. Но в строке “c” запись не производилась, поэтому читается код FFH, и, следовательно, за достоверные данные принимается не строка “b”, а строка “a”.

АЛГОРИТМ УСТАНОВКИ ФЛАГОВ ДОСТОВЕРНОСТИ И НЕДОСТОВЕРНОСТИ ПРИ РАБОТЕ С БЛОКАМИ ПАМЯТИ

Как было описано выше, для того, чтобы зафиксировать текущее состояние блока данных, используются два флага – ФД и ФН. Рассмотрим алгоритм установки и снятия флагов на примере двух блоков, имеющих номера “n” “n+1”.

Выполнение операций по установке флагов ФД и ФН приводится ниже.

Шаг 1 Начальное состояние

Блок	n	n+1
ФД	FFH	FFH
ФН	FFH	FFH
Данные	FFH	FFH
Данные	FFH	FFH
Данные	FFH	FFH

Выполняется поиск первого из блоков, предназначенных для хранения данных. Проверяется, что в этот блок не производилась запись. Читается флаг ФД и если там записан код FFH, то этот блок выбирается для дальнейшего хранения данных.

Шаг 2 Запись кода 00 в блок “n” устанавливает этот блок как разрешенный для записи

Блок	n	n+1
ФД	00H	FFH
ФН	FFH	FFH
Данные	FFH	FFH
Данные	FFH	FFH
Данные	FFH	FFH

Шаг 3 Запись данных в блок “n”

Блок	n	n+1
ФД	00H	FFH
ФН	FFH	FFH
Данные	**Data**	FFH
Данные	FFH	FFH
Данные	FFH	FFH

Данные сохраняются последовательно в разрешенный блок.

Если блок становится полным, то выполняется поиск другого блока для того, чтобы сохранить последующие данные.

Шаг 4 Блок “n” заполнен данными

Блок	n	n+1
ФД	00H	FFH
ФН	FFH	FFH
Данные	**Data**	FFH
Данные	**Data**	FFH
Данные	**Data**	FFH

Шаг 5 Запись данных продолжается в блоке “n+1”

Блок	n	n+1
ФД	00H	FFH
ФН	FFH	FFH
Данные	**Data**	**Data**
Данные	**Data**	FFH
Данные	**Data**	FFH

Когда блок будет найден, то в него будут записаны новые кадры данных.

После того, как передача данных закончена, устанавливается флаг ФД. Установка этого флага делает блок “n+1” разрешенным для записи.

Шаг 6 В блоке “n+1” устанавливается флаг разрешения для записи

Блок	n	n+1
ФД	00H	00H
ФН	FFH	FFH
Данные	**Data**	**Data**
Данные	**Data**	FFH
Данные	**Data**	FFH

Шаг 7 В блоке “n” снимается флаг разрешения для записи

Блок	n	n+1
ФД	00H	00H
ФН	00H	FFH
Данные	**Data**	**Data**
Данные	**Data**	FFH
Данные	**Data**	FFH

После того, как установка флага ФД делает блок “n+1” разрешенным для записи, производится сброс флага ФН в блоке “n” – и он становится запрещенным для записи.

Шаг 8 Блок “n+1” заполнен данными

Блок	n	n+1
ФД	00H	00H
ФН	00H	FFH
Данные	**Data**	**Data**
Данные	**Data**	**Data**
Данные	**Data**	**Data**

Когда блок “n+1” заполняется данными, программа переходит к стиранию содержимого блока “n”.

Шаг 9 Блок “n” – проводится стирание всего блока данных

Блок	n	n+1
ФД	FFH	00H
ФН	FFH	FFH
Данные	FFH	**Data**
Данные	FFH	**Data**
Данные	FFH	**Data**

Шаг 10 Блок ”n“ – проводится запись последнего слова данных

Блок	n	n+1
ФД	FFH	00H
ФН	FFH	FFH
Данные	**Data**	**Data**
Данные	FFH	**Data**
Данные	FFH	**Data**

Шаг 11 В блоке “n” устанавливается флаг разрешения для записи

Блок	n	n+1
ФД	00H	00H
ФН	FFH	FFH
Данные	**Data**	**Data**
Данные	FFH	**Data**
Данные	FFH	**Data**

После того, как передача данных закончена, устанавливается флаг ФД. Установка этого флага делает блок «n» разрешенным для записи.

Шаг 12 В блоке “n+1” снимается флаг разрешения для записи

Блок	n	n+1
ФД	00H	00H
ФН	FFH	00H
Данные	**Data**	**Data**
Данные	FFH	**Data**
Данные	FFH	**Data**

После того, как установка флага ФД делает блок “n” разрешенным для записи, производится сброс флага ФН в блоке “n+1” – и он становится запрещенным для записи.

Далее следует стирание содержимого блока “n+1” и цикл работы устройства повторяется.

УСЛОВИЯ, НЕОБХОДИМЫЕ ДЛЯ ВЫПОЛНЕНИЯ РЕЖИМА ЭМУЛЯЦИИ EEPROM

Для правильной работы микроконтроллера в режиме эмуляции EEPROM необходимо выполнить условия, перечисленные в табл. 3.

Таблица 3 Условия, необходимые для правильной работы микроконтроллера в режиме эмуляции EEPROM

Условия, которые необходимо выполнить	Описание
Необходимо иметь достаточный запас по глубине стека.	При работе пользовательской программы часть области стека будет уже занята. Необходимо, чтобы при работе программы эмуляции EEPROM был достаточный запас по глубине стека, чтобы не произошло переполнение или опустошение стека, иначе это может привести к непредсказуемым результатам. В Л[3] указаны требования по глубине стека, необходимые для программы эмуляции EEPROM
Должна быть выделена оперативная память, требуемая для программы эмуляции EEPROM	Для работы программы эмуляции EEPROM необходимо выделить во внутренней высокоскоростной оперативной памяти специальную область, которая будет использоваться как буфер данных. Область должна быть защищена как оперативная память, выделенная для эмуляции EEPROM, где происходит чтение, и данные записи сохранены временно. В дополнение к оперативной памяти для программы, описанной слева, только область стека используется в соответствии с программой эмуляции EEPROM
Работа сторожевого таймера (WDT)	Поскольку при работе эмуляции EEPROM невозможно выполнять обработку других задач, то это может привести к срабатыванию WDT. Поэтому установите время срабатывания WDT на 10 мс или на больший период времени так, чтобы во время работы программы эмуляции не произошло срабатывание WDT
Запретите сброс	Не делайте сброс микроконтроллера в течение операций программы эмуляции EEPROM. Когда происходит сброс, любые данные во флэш-памяти, к которой производилось обращение, становятся неопределенными
Запретите отключение напряжение питания и/или прерывание	Убедитесь, что в течение операций программы эмуляции EEPROM на микроконтроллер подается стабильное напряжение питания. Когда происходит отключение питания или происходят прерывания, любые данные во флэш-памяти, к которой производилось обращение, становятся неопределенными

Полное описание процедур эмуляции EEPROM приведено в Л[3]. Даны коды программ, что позволяет значительно сократить время на изучение и освоение данного способа работы.

ЗАКЛЮЧЕНИЕ

Приведенный в данной статье способ записи данных позволяет проводить эмуляцию EEPROM во FLASH памяти. Объем записываемых данных и число циклов перезаписи определяется типом используемого микроконтроллера и зависит от объема внутренней памяти микроконтроллера и гарантируемого производителем числа циклов перезаписи.

Литература

1. AVR105: Power Efficient High Endurance Parameter Storage in Flash Memory
2. 8-bit Microcontroller with 128K Bytes of ISP Flash and CAN Controller; AT90CAN128 Automotive.
3. 78K0S/Kx1+ 8-Bit Single-Chip Microcontrollers EEPROMTM Emulation; Preliminary Application Note; U17379EJ1V0AN00 http://www.ee.nec.de.