УДК 681.3.06(07)

Лосихин Д.А.

ОБМЕН ДАННЫМИ МИКРОКОНТРОЛЛЕРОВ С ВНЕШНЕЙ ЭНЕРГОНЕЗАВИСИМОЙ ПАМЯТЬЮ ПО ДВУХПРОВОДНОМУ ПОСЛЕДОВАТЕЛЬНОМУ ИНТЕРФЕЙСУ С ПРОВЕРКОЙ ЦЕЛОСТНОСТИ ПО АЛГОРИТМУ 8-БИТНОГО ЦИКЛИЧЕСКОГО ИЗБЫТОЧНОГО КОДА CRC-8/MAXIM

ГВУЗ «Украинский государственный химико-технологический университет», г. Днепр, Украина

Статья посвящена организации последовательной связи между микроконтроллером и внешним программируемым постоянным запоминающим устройством с контролем целостности данных. Получение достоверной информации по информационно-измерительным каналам в системах автоматизации в большей мере обеспечивается интеллектуальными датчиками, основными аппаратными элементами которых являются первичные измерительные преобразователи, микроконтроллер, энергонезависимая память и интерфейсные модули. На программном уровне, вычислительные возможности микроконтроллера позволяют реализовать в интеллектуальных датчиках алгоритмы первичной обработки информации, диагностического самоконтроля и алгоритмы адаптации к изменяющимся внешним воздействиям. Во внешней энергонезависимой памяти при этом находятся данные калибровки для коррекции погрешности измерения, параметры алгоритма самовосстановления при возникновении единичного дефекта, параметры алгоритмов самообучения. Однако риск получения недостоверного результата измерения зависит и от состояния самих аппаратных средств интеллектуального датчика, в частности, от выносливости энергонезависимой памяти, и от ошибок передачи данных по цифровым интерфейсным каналам, в частности, по последовательному интерфейсу связи между микроконтроллером и энергонезависимой памятью. Для повышения надёжности передачи и хранения данных в носителях информации применяется избыточное кодирование с использованием схем и алгоритмов вычисления циклического избыточного кода (CRC). Поддержку вычисления CRC на аппаратном уровне обеспечивают не все микросхемы энергонезависимой памяти, да и микроконтроллеры, к сожалению, не имеют таких аппаратных кодеров контроля ошибок в потоке последовательных данных. Разрешение проблемы на программном уровне рассматривается на примере обмена данными микроконтроллера ATmega328p (Microchip Technology Inc.) с внешней энергонезависимой памятью AT24C04C/AT24C08C (Microchip Technology Inc.) по двухпроводному последовательному интерфейсу I2C (Two-Wire) с проверкой целостности по алгоритму 8-и битного циклического избыточного кода CRC-8 (Maxim Integrated); разработаны соответствующие библиотеки функций на языке C и тестовая программа; код проверен на реальных устройствах.

Ключевые слова: измерительная система, интеллектуальный датчик, микроконтроллер, внешнее электрически стираемое программируемое постоянное запоминающее устройство, последовательная синхронная ассиметричная шина связи интегральных микросхем, контроль целостности данных, циклический избыточный код, программирование на языке C.

DOI: 10.32434/2521-6406-2019-6-2-23-34

Анализ литературных данных и постановка проблемы

Хранение данных во внешней электрически стираемой энергонезависимой памяти сопряжено с выносливостью микросхем памяти, отчасти зависящей от производителя, отчасти от условий эксплуатации. Также, возможны ошибки программиста при организации взаимодействия микроконтроллеров с модификациями микросхем внешней памяти по соответствующему техническому описанию.

Связь с микросхемами электрически стираемой перепрограммируемой постоянной памяти (ЭСППЗУ; EEPROM) организована производителями микросхем на основе последовательных интерфейсов: 1-Wire (Dallas Semiconductor – Maxim Integrated); CAN (Robert Bosch GmbH); I2C (Philips Semiconductors – NXP Semiconductors), Microwire (National Semiconductor – Texas Instruments Incorporated), SPI (Atmel – Microchip Technology Inc.), и др. [1]. Каждый из интерфейсов обладает особенностями, позволяющими эффективно решать поставленную задачу взаимодействия компонентов измерительных систем в информационно-управляющих системах.

Отличительной особенностью однопроводного интерфейса последовательной EEPROM Maxim 1-Wire является наличие функции CRC (Cyclic redundancy check) проверки целостности данных на аппаратном уровне. Например, встроенная в аппаратные средства функция 8-битной проверки, представленная схемой сдвигового регистра с обратными связями (рис. 1), используется для проверки уникального 64-битного ROM кода, записанного в каждом 1-Wire устройстве [2]. Значение CRC вычисляется для 56 бит ROM кода и записывается в его самый старший байт.

При чтении данных по протоколу 1-Wire, ведущее устройство реализует аналогичную функцию CRC проверки целостности данных на программном уровне, сравнивая эталонное и расчётное значения CRC для принятых данных.

С учётом того, что двухпроводная шина I2C не предназначена для удалённой связи устройств, последовательная I2C-совместимая EEPROM у разных производителей не имеет аппаратную функцию проверки целостности данных. Однако, возможны структуры измерительных систем, в которых интеллектуальный датчик, и его основные составные модули: первичные измерительные преобразователи, микроконтроллер и внешняя память находятся в условиях влияния значительных внешних помех, и значения измеряемого параметра обрабатываются микроконтроллером с учётом констант или переменных из внешней I2C-совместимой EEPROM. И хотя целостность данных в I2C устройствах поддерживается встроенными фильтрами для подавления помех [3], всё же, для повышения надёжности передачи и хранения информации необходима проверка целостности данных, которую можно обеспечить на программном уровне.

Формулирование целей статьи

Обеспечить проверку целостности данных при обмене информацией микроконтроллера и внешней электрически стираемой перепрограммируемой постоянной памятью по двухпроводному последовательному интерфейсу на уровне приложения.

Для достижения поставленной цели необходимо:

– организовать взаимодействие между микроконтроллером (ведущее устройство) и ЭСППЗУ (ведомое устройство) с контролем целостности вещественных переменных;

– разработать программное обеспечение на примере взаимодействия AVR-микроконтроллера ATmega328p [4] и стандартной I2C-совместимой EEPROM AT24C04C [5].

Изложение основного материала исследования

Взаимодействие между микроконтроллером и внешней EEPROM с контролем целостности вещественных переменных на уровне приложения осуществляется через буфер объединения типов данных в оперативной памяти (RAM) микроконтроллера. Буфер предоставляет в памяти RAM одно и то же пространство для ве-

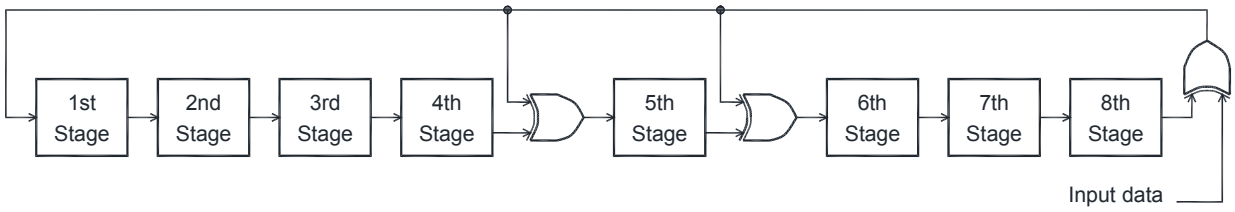

Рис. 1. Maxim 1-Wire 8-битный CRC

щественной переменной типа float (m=4 байта), беззнаковой целой переменной типа unsigned int (m=4 байта) и массиву из 5-ти беззнаковых целых элементов типа byte (m+1=5 байт) [6].

При записи вещественной переменной во внешнюю EEPROM (рис. 2), из RAM вещественная переменная типа float записывается в буфер RAM buffer; по ссылке на эту область памяти создается копия беззнаковой целой переменной uint. По ссылке на копию беззнаковой целой переменной uint имеем доступ к каждому байту этой переменной. Последовательность этих байт используется в функции расчёта 8-битного CRC, который помещается в последний байт буфера RAM buffer. Затем, из буфера, в соответствии с протоколом обмена I2C [4,5], данные data bytes и значение CRC, через соответствующий порт ввода/вывода I/O, последовательно от младшего байта к старшему, записываются по заданному адресу во внешнюю память external EEPROM.

Чтение байт данных и 8-битного значения CRC из внешней EEPROM по протоколу обмена I2C происходит в том же порядке, что и при записи (рис. 3). Считываемые данные data bytes и CRC помещаются в буфер RAM buffer; по ссылке на эту область памяти создается копия беззнаковой целой переменной uint. По ссылке на копию беззнаковой целой переменной uint имеем доступ к каждому байту этой переменной. Последовательность этих байт используется в функции расчёта и проверки 8-битного CRC, который сравнивается с последним байтом буфера RAM buffer; в случае их равенства функция возвращает значение флага flag равным нулю, что указывает на целостность принятых данных.

Дальнейшие действия в программе должны быть переданы управляющему оператору ветвления или оператору цикла с проверкой условия равенства нулю значения флага flag; если значение флага равно нулю, из буфера RAM buffer считывается проверенное значение вещественной переменной; если флаг не равен нулю, то целостность данных не подтверждается и возможно организовать повторное считывание данных из внешней EEPROM по этому же адресу. В тестовой программе (листинг 1) в коммуникационный порт выводятся рассчитанные значение CRC перед записью вещественной переменной во внешнюю EEPROM и значение флага после чтения вещественной переменной из внешней EEPROM по одному адресу.

Функции расчёта и проверки 8-битного CRC: calc CRC и check CRC основаны на математической модели циклического избыточного контроля [7].

Контрольные биты определяются, как двоичный код полинома R(x) остатка от деления в поле Галуа GF(2) полинома исходных данных M(x) степени $<n$, умноженного на x^n, на порождающий полином G(x) степени n:

$$M(x)\cdot x^n=G(x)\cdot Q(x)+R(x), \qquad (1)$$

где Q(x) – частное от деления полинома.

Для расчёта CRC-8/MAXIM следует учесть спецификации:

– число избыточных бит, n=8;

– отражённый, т.е. обратный порядок представления входных данных и результата;

– порождающий полином, $G(x)=x^8+x^5+x^4+1$;

– начальное значение CRC равно

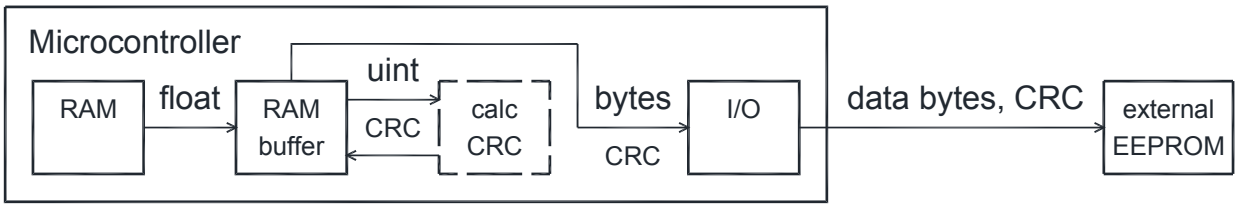

Рис. 2. Запись вещественной переменной во внешнюю EEPROM

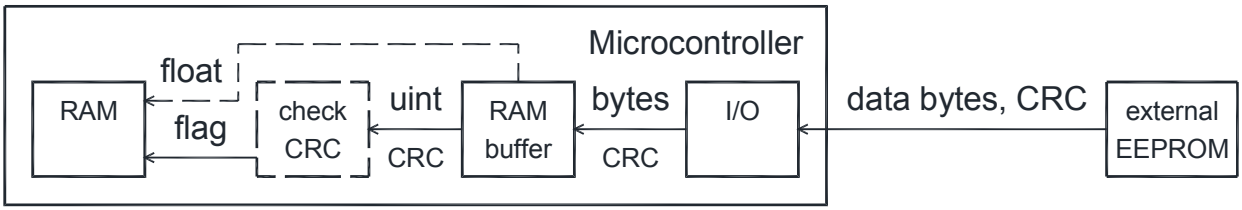

Рис. 3. Чтение вещественной переменной из внешней EEPROM

$R_0(x)$=0x00 (при алгоритмизации).

Пусть задан полином исходных данных, например, $M(x)=x^7+x^6+x^5+x^3+x^2$, и отражённый полином равен $M(x)=refl[M(x)]=x^5+x^4+x^2+x+1$. В этом случае, расчёт полинома CRC-8/MAXIM можно представить в виде:

$$\frac{x^{13}+x^{12}+x^{10}+x^9+x^8}{x^8+x^5+x^4+1}= \\ =\left(x^5+x^4+x\right)+\left(x^6+x^4+x\right). \quad (2)$$

В результате расчёта (2) получаем в остатке отражённый полином $R(x)=x^6+x^4+x$, и, окончательно, полином CRC $R(x)=refl[R(x)]=x^6+x^3+x$.

Такой же результат, получим при делении двоичных кодов в поле Галуа GF(2), табл. 1.

Последовательная схема (рис. 1), деления двоичного кода исходных данных в регистре CRC на двоичный код полинома $G(x)=x^8+x^5+x^4+1$ решает эту задачу на аппаратном уровне. На программном уровне задача решается аналогично с использованием разработанных библиотечных функций (листинги 2–7).

Выводы

В приведенном здесь решении вычисляется 8-битный CRC для вещественной переменой, занимающей в памяти 4 байта. Избыточность кодирования составляет 0,2, а эффективное использование пространства EEPROM составляет 75% с учётом 16-байтной страничной адресации памяти.

С целью более надёжного контроля целостности данных можно вычислять 8-битный CRC

Таблица 1

Расчёт значения CRC-8/MAXIM

Этапы цикла	Позиция бита	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
	Исходные данные									1	1	1	0	1	1	0	0
0	Отражение исходных данных и умножение на x^8	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0
1	Сдвиг данных в регистре CRC, (рис. 1), и XOR с порождающим полиномом			1	0	0	1	1	0	0	0	1					
2	Результат XOR в регистре CRC			0	1	0	0	0	1	0	0	1					
1	Сдвиг и XOR				1	0	0	1	1	0	0	0	1				
2	Результат XOR				0	0	0	1	0	0	0	1	1				
1	Сдвиг и XOR							1	0	0	1	1	0	0	0	1	
2	Результат XOR							0	0	0	1	0	1	0	0	1	0
3	Отражение результата в регистре CRC									0	1	0	0	1	0	1	0

Таблица 2

Результаты расчёта CRC для некоторых вариантов исходных данных

Значение переменной типа float	Расчётное значение CRC-8/MAXIM	Содержимое буфера обмена перед записью в EEPROM	Тестовое содержимое буфера обмена после записи в EEPROM	Содержимое буфера обмена после чтения из EEPROM	Значение флага целостности считанных данных из EEPROM
3,1415	0x4A	0x56 0x0E 0x49 0x40 0x4A	0x00 0x00 0x00 0x00 0x00	0x56 0x0E 0x49 0x40 0x4A	0x00
6,2831	0x73	0x28 0x0F 0xC9 0x40 0x73	0x00 0x00 0x00 0x00 0x00	0x28 0x0F 0xC9 0x40 0x73	0x00
0,61803	0xA5	0x37 0x37 0x1E 0x3F 0xA5	0x00 0x00 0x00 0x00 0x00	0x37 0x37 0x1E 0x3F 0xA5	0x00
1,4142	0xCA	0x81 0x04 0xB5 0x3F 0xCA	0x00 0x00 0x00 0x00 0x00	0x81 0x04 0xB5 0x3F 0xCA	0x00
2,1782	0x9C	0xA1 0x67 0x0B 0x40 0x9C	0x00 0x00 0x00 0x00 0x00	0xA1 0x67 0x0B 0x40 0x9C	0x00

Результаты расчёта

Листинг 1

Основная тестовая программа «lda_lab_AT24C04_main.c»

```
#define F_CPU 16000000UL // Рабочая частота AVR-микроконтроллера
#include <avr/io.h>
#include <stdint.h>
#include "lda_lab_UART.h" // Заголовочный файл библиотеки UART
#include "lda_lab_i2c.h"  // Заголовочный файл библиотеки I2C
#include "lda_lab_AT24C04.h" // Заголовочный файл библиотеки EEPROM AT24C04C
// Буфер обмена информацией с внешней EEPROM
float2bytes_t *var;
// Адрес памяти данных внешней EEPROM.
// Выбор произвольный в пределах допустимого пространства памяти;
// здесь выбран последний возможный адрес в EEPROM AT24C04C
// (device adress 0x50) для переменной типа float2bytes_t
uint8_t adress = 0xFB;
int main(void){
  USART_Init(); // Инициализация модуля USART
  initI2C(); // Инициализация модуля TWI
  uint8_t crcf; // Расчётное значение crc-8-maxim
  uint8_t crc_flag; // Флаг целостности данных (0x00 – подтверждение целостности)
  float arr[6] = {0}; // Массив (n+1) тестовых исходных данных (n) для записи/чтения
  // Варианты тестовых исходных данных сличены с расчётом в приложении на сайте:
  // https://gregstoll.com/~gregstoll/floattohex/
  arr[0] = 3.1415; // 0x40490E56
  arr[1] = 6.2831; // 0x40c90f28
  arr[2] = 0.61803; // 0x3f1e3737
  arr[3] = 1.4142; // 0x3fb50481
  arr[4] = 2.1782; // 0x400b67a1
  // Основные выражения программы далее пронумерованы в комментариях:
  // 1. Вычисляем CRC вещественного числа по адресу (arr+4) и помещаем результат в буфер var
  crcf = crc_8_maxim(arr+4);
  // Тест. Вывод численного значения CRC на консоль
  print_byte_HEX(crcf);
  // 2. Записываем в EEPROM вещественное число – 4 байта и CRC – 1 байт
  five_reg_w(ADRESS_W_AT24C04_0x50, adress, var);
  //даём время EEPROM (зависит от производителя устройства) привести себя в порядок
  _delay_ms(15);
  // 3. Считываем из внешней EEPROM вещественное число и CRC в буфер var
  five_reg_r(ADRESS_W_AT24C04_0x50, ADRESS_R_AT24C04_0x50, adress, var);
  // 4. Проводим проверку на целостность считанных из внешней EEPROM данных;
  crc_flag = crc_8_maxim_check(var);
    // выводим на консоль значение флага:
  print_byte_HEX(crc_flag);
  return;
}
```

Листинг 2

Библиотека «lda_lab_AT24C04.c» функций взаимодействия микроконтроллера с внешней энергонезависимой I2C-совместимой EEPROM AT24C04/AT24C08

```
#include <stdint.h>
#include "lda_lab_AT24C04.h"
#include "lda_lab_i2c.h"
extern float2bytes_t *var;
void five_reg_w(uint8_t dev_w_adress, uint8_t reg_adress, float2bytes_t* data){
  startI2C();
  sendI2C(dev_w_adress); //ADRESS_W_AT24C04_0x50
  sendI2C(reg_adress);
  sendI2C(data->b[0]);
  sendI2C(data->b[1]);
  sendI2C(data->b[2]);
  sendI2C(data->b[3]);
  sendI2C(data->b[4]);
  stopI2C();
}
void five_reg_r(uint8_t dev_w_adress, uint8_t dev_r_adress, uint8_t reg_adress, float2bytes_t* data){
  startI2C();
  sendI2C(dev_w_adress); //ADRESS_W_AT24C04_0x50
  sendI2C(reg_adress);
  startI2C();
  sendI2C(dev_r_adress); //ADRESS_R_AT24C04_0x50
  data->b[0] = readACKI2C();
  data->b[1] = readACKI2C();
  data->b[2] = readACKI2C();
  data->b[3] = readACKI2C();
  data->b[4] = readNACKI2C();
  stopI2C();
}
uint8_t crc_8_maxim(float *arr){ //arr: обасть видимости main
    uint8_t n = 8; //bits
    uint8_t m = 4; //sizeof(float)
    int i, j;
    uint8_t lsb = 0;
    uint8_t crc = 0;

    var = (float2bytes_t *)arr; //var: глобальная переменная
    float2bytes_t r; //r: локальная переменная
    r.b4 = var->b4; //
    for(j = 0; j < m; j++){
        crc ^= r.b[j];
        for(i = 0; i < n; i++){
            lsb =  crc & 1; //передвигаем 8-битное окно на младший байт и определяем LSB
            crc >>= 1;
            if(lsb) crc ^= POLINOM; //глобальная константа
        }
        r.b[j] = crc;
    }
    return var->b[4] = crc; //записываем crc в пятый байт глобальной переменной типа float2bytes_t
}
uint8_t crc_8_maxim_check(float2bytes_t *var){ //var: глобальная переменная
    uint8_t n = 8; //bits
    uint8_t m = 4; //bytes: sizeof(float)
    int i, j;
    uint8_t lsb = 0;
    uint8_t crc = 0;
    float2bytes_t r; //r: локальная переменная
    r.b4 = var->b4;
    for(j = 0; j < m; j++){
        crc ^= r.b[j];
        for(i = 0; i < n; i++){
            lsb =  crc & 1; //передвигаем 8-битное окно на младший байт и определяем LSB
            crc >>= 1;
            if(lsb) crc ^= POLINOM; //глобальная константа
        }
        r.b[j] = crc;
    }
    return var->b[4] != crc;
}
```

Листинг 3

Заголовочный файл библиотеки «lda_lab_AT24C04.h»

```
#include <stdint.h>
//полиномальная функция для crc-8-maxim
#define POLINOM 0b10001100
#define float2bytes_t union float_to_bytes
// Адреса AT24C04C, [2] – 6.1.1 AT24C04C Device Addressing,
// на чтение:
#define ADRESS_R_AT24C04_0x50 0b10100001  // 1010 – Device Address
                                // 00 – hardwired device address
                                // 00 – Most Significant Bit of the Word Address
                                // 1 – R/W Select bit
#define ADRESS_R_AT24C04_0x51 0b10100011  // 1010 – Device Address
                                // 00 – hardwired device address
                                // 01 – Most Significant Bit of the Word Address
                                // 1 – R/W Select bit
// на запись:
#define ADRESS_W_AT24C04_0x50 0b10100000  // 1010 – Device Address
                                // 00 – hardwired device address
                                // 00 – Most Significant Bit of the Word Address
                                // 0 – R/W Select bit
#define ADRESS_W_AT24C04_0x51 0b10100010  // 1010 – Device Address
                                // 00 – hardwired device address
                                // 01 – Most Significant Bit of the Word Address
                                // 0 – R/W Select bit
// Тип буфера обмена данными с внешней EEPROM
float2bytes_t {
                        float fb;
                        uint32_t b4;
                        uint8_t b[5];
};
// ОСНОВНЫЕ ФУНКЦИИ ОБМЕНА ДАННЫМИ с EEPROM AT24C04C по I2C
//---------------------------------------------------------
// Запись в пять регистров AT24C04C, fig. 7-2
// 7.2 Page Write (написал, проверил)
void five_reg_w(uint8_t dev_w_adress, uint8_t reg_adress, float2bytes_t* data);
//---------------------------------------------------------
// Чтение пяти регистров из AT24C04C, fig. 8-3
// 8.3 Sequential Read (написал, проверил)
void five_reg_r(uint8_t dev_w_adress, uint8_t dev_r_adress, uint8_t reg_adress, float2bytes_t* data);
//---------------------------------------------------------
// ОСНОВНЫЕ ФУНКЦИИ CRC-8-MAXIM в EEPROM AT24C04C
//---------------------------------------------
// Функция вычисляет и возвращает значение crc-8-maxim:
uint8_t crc_8_maxim(float *arr);
//------------------------------
// Проверка crc-8-maxim; функция возвращает значение флага проверки:
uint8_t crc_8_maxim_check(float2bytes_t *var);
//---------------------------------------------
```

Листинг 4

Библиотека «lda_lab_i2c.c» функций взаимодействия AVR-микроконтроллера с периферией по шине I2C

```
#include <avr/io.h>
#include "lda_lab_i2c.h"
void initI2C(){
   TWBR = (F_CPU/F_SCL-16)/2/PRESCALER_TWI;
   //предделитель PrescalerValue = 4
   TWSR &= ~(1<< TWPS1);
   TWSR |= (1<< TWPS0);
   //или по умолчанию  PrescalerValue =1
   //TWSR &= ~(1<<TWPS1)&~ (1<<TWPS0);
   //разрешаем функционирование TWI
   TWCR |= (1<<TWEN)|(1<<TWIE);
}
void waitForCompleteI2C(void) {
   while(~TWCR&(1<<TWINT)); //ждём пока не появится 1
}
void startI2C(void) {
   TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTA);
   waitForCompleteI2C();
}
void stopI2C(void) {
   TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
}
uint8_t readACKI2C(void) {
   TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
   waitForCompleteI2C();
  return (TWDR);
}
uint8_t readNACKI2C(void) {
   TWCR = (1<<TWINT)|(1<<TWEN);
   //TWCR &= ~(1<<TWEA); //необязательно
   waitForCompleteI2C();
  return (TWDR);
}
void sendI2C(uint8_t data) {
   TWDR = data;
   TWCR = (1<<TWINT)|(1<<TWEN);
   waitForCompleteI2C();
}
```

Листинг 5

Заголовочный файл библиотеки «lda_lab_i2c.h»

```
#include <avr/io.h>
#ifndef F_CPU
// Рабочая частота контроллера, Гц
#define F_CPU 16000000UL
#endif
// Частота обмена информацией по I2C, Гц
#define F_SCL 100000
// Предделитель, [2] – 26.5.2. Bit Rate Generator Unit
#define PRESCALER_TWI 4
// ОСНОВНЫЕ ФУНКЦИИ ИНТЕРФЕЙСА I2C
//------------------------------
// Инициализация скорости шины 100 кГц (F_CPU = 16 МГц)
void initI2C(void);
//----------------
// Ждать, пока устройство установит TWINT флаг
void waitForCompleteI2C(void);
//----------------------------
// Отправить старт-условие (устанавливаем TWSTA)
void startI2C(void);
//-----------------
// Отправить стоп-условие (устанавливаем TWSTO)
void stopI2C(void);
//----------------
// Загрузить данные, отправить их, и ждать завершения
void sendI2C(uint8_t data);
//-------------------------
// Считать от ведомого, передать ACK по завершению (установить TWEA)
uint8_t readACKI2C(void);
//----------------------
// Считать от ведомого, передать NOACK по завершению (не устанавливать TWEA)
uint8_t readNACKI2C(void);
//----------------------
```

Листинг 6

Библиотека «lda_lab_UART.c» функций взаимодействия AVR-микроконтроллера с периферией по стандартному последовательному коммуникационному протоколу

```
#include "lda_lab_UART.h"
void USART_Init(void) {
  UBRR0H = UBRR_VALUE >> 8;
  UBRR0L = UBRR_VALUE;
  UCSR0B |= (1<<TXEN0)|(1<<RXEN0);
  UCSR0C &= ~(1<<USBS0); // один стоп-бит
  UCSR0C &= ~(1<<UPM01)&~(1<<UPM00); //Нет проверки на чётность
  UCSR0C &= ~(1<<UMSEL01)&~(1<<UMSEL00)&~(1<<UCPOL0); //асинхронный режим
  UCSR0C |= (1<< UCSZ01)|(1<< UCSZ00); //Устанавливем формат данных: 8 бит
  UCSR0B &= ~(1<<UCSZ02); //Устанавливем формат данных: 8 бит
}
void USART_Transmit(uint8_t ch) {
  while(!(UCSR0A & (1 << UDRE0)));   // Ожидаем когда очистится буфер передачи
  UDR0 = ch; // Помещаем данные в буфер, начинаем передачу
}
uint8_t USART_Receive(void) {
  while (!(UCSR0A & (1 << RXC0))) ; //ждём пока появятся новые данные
  return UDR0;
}
void print_byte_HEX(uint8_t data){
  uint8_t tmp = 0;
  tmp = (data >> 4);
  if(tmp > 9)
    USART_Transmit( tmp + 0x37 );
  else USART_Transmit( tmp + 0x30 );
  tmp = (data & 0x0f);
  if(tmp > 9)
    USART_Transmit( tmp + 0x37 );
  else USART_Transmit( tmp + 0x30 );
  USART_Transmit( 0x0A );
  USART_Transmit( 0x0D );
  return;
}
```

Листинг 7

Заголовочный файл библиотеки «lda_lab_UART.h»

```
#include <avr/io.h>
#ifndef F_CPU
#define F_CPU 16000000UL // Рабочая частота контроллера
#endif
#define BAUD_RATE 9600UL
#define UBRR_VALUE ((F_CPU/(16*BAUD_RATE))-1)
// ОСНОВНЫЕ ФУНКЦИИ UART
//----------------------
// Инициализация модуля AVR USART
void USART_Init(void);
//----------------------
// Функция передачи байта
void USART_Transmit(uint8_t ch);
//-------------------------------
// Функция приёма байта
unsigned char USART_Receive(void);
//--------------------------------
// Функция выводит символы в кодировке ASCII через USART в COM-порт
void print_byte_HEX(uint8_t data);
//--------------------------------
```

для каждого байта вещественной переменной. При этом избыточность кодирования увеличится и будет равна 0,5, но приблизительно в два раза увеличится время записи/чтения данных EEPROM, а эффективное использование пространства EEPROM уменьшится до 50%.

Вычисление 16-битного значения CRC так же приведёт к усложнению алгоритма расчёта, при этом избыточность кодирования увеличится и будет равна 0,33, а эффективное использование пространства EEPROM уменьшится до 50%.

Результаты расчёта по тестовым исходным данным подтверждают верное решение поставленной задачи. Разработанные библиотечные функции (листинг 2–7) могут быть применены конкретно для микросхем памяти AT24C04C/AT24C08C, и использованы, как шаблоны, для других модификаций EEPROM AT24C, поддерживающих двухпроводной последовательный интерфейс I2C, с учетом соответствующих организаций карт памяти и адресации.

СПИСОК ЛИТЕРАТУРЫ

1. *Лапин А.А.* Интерфейсы. Выбор и реализация. – М.: Техносфера, 2005. – 168 с.

2. Understanding and Using Cyclic Redundancy Checks with Maxim 1-Wire and iButton Products (URL: https://pdfserv.maximintegrated.com/en/an/AN27.pdf).

3. Интерфейсы периферийных устройств / А.О. Ключев, Д.Р. Ковязина., Е.В. Петров, А.Е. Платунов. – СПб.: СПбГУ ИТМО, 2010. – 290 с.

4. ATmega328P. DATASHEET (URL: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf).

5. AT24C04C and AT24C08C. DATASHEET (URL: http://ww1.microchip.com/downloads/en/devicedoc/atmel-8787-seeprom-at24c04c-08c-datasheet.pdf).

6. Chopra R. C Programming: A Self-Teaching Introduction. New Delhi // New Age International Publishers, 2018.

7. *Авдеев В.А.* Периферийные устройства: интерфейсы, схемотехника, программирование. – М.: ДМК Пресс, 2009. – 848 с.

Поступила в редакцию 13.11.2019

ОБМІН ДАНИМИ МІКРОКОНТРОЛЕРІВ З ЗОВНІШНЬОЮ НЕЗАЛЕЖНОЮ ПАМ'ЯТТЮ ПО ДВОПРОВІДНОМУ ПОСЛІДОВНОМУ ІНТЕРФЕЙСУ З ПЕРЕВІРКОЮ ЦІЛІСНОСТІ ЗА АЛГОРИТМОМ 8-БІТНОГО ЦИКЛІЧНОГО НАДЛИШКОВОГО КОДУ CRC-8/MAXIM

Лосіхін Д.А.

Стаття присвячена організації послідовного зв'язку між мікроконтролером і зовнішнім програмувальним постійним запам'ятовуючим пристроєм з контролем цілісності даних. Отримання достовірної інформації з інформаційно-вимірювальних каналів в системах автоматизації в більшою мірою забезпечується інтелектуальними датчиками, основними апаратними елементами яких є первинні вимірювальні перетворювачі, мікроконтролер, енергонезалежна пам'ять та їх інтерфейсні компоненти. На програмному рівні, обчислювальні можливості мікроконтролера дозволяють реалізувати в інтелектуальних датчиках алгоритми первинної обробки інформації, діагностичного самоконтролю і алгоритми адаптації до постійно змінюваних зовнішніх впливів. У зовнішній незалежній пам'яті при цьому знаходяться дані калібрування для корекції похибки вимірювання, параметри алгоритму самовідновлення при виникненні одиничного дефекту, параметри алгоритмів самонавчання. Однак ризик отримання недостовірного результату вимірювання залежить також і від стану самих апаратних засобів інтелектуального датчика, зокрема, від витривалості енергонезалежної пам'яті, і від помилок передачі даних з цифрових інтерфейсних каналах, зокрема, по послідовному інтерфейсу зв'язку між мікроконтролером і енергонезалежною пам'яттю. Для підвищення надійності передачі і зберігання даних в носіях інформації застосовується надлишкове кодування з використанням схем і алгоритмів обчислення циклічного надлишкового коду (CRC). Підтримку обчислення CRC на апаратному рівні забезпечують не всі мікросхеми енергонезалежної пам'яті, та й мікроконтролери, на жаль, не мають таких апаратних кодерів контролю помилок в потоці послідовних даних. Вирішення проблеми на програмному рівні розглядається на прикладі обміну даними мікроконтролера ATmega328p (Microchip Technology Inc.) з зовнішньою енергонезалежною пам'яттю AT24C04C/AT24C08C (Microchip Technology Inc.) по двопровідному послідовному інтерфейсу I2C (Two-Wire) з перевіркою цілісності за алгоритмом 8-бітного циклічного надлишкового коду CRC-8 (Maxim Integrated); розроблені відповідні бібліотеки функцій на мові С і тестова програма; код перевірений на реальних пристроях.

Ключові слова: вимірювальна система, інтелектуальний датчик, мікроконтролер, зовнішній постійний запам'ятовувальний пристрій, що програмується за допомогою електрики, послідовна синхронна асиметрична шина зв'язку інтегральних мікросхем, контроль цілісності даних, циклічний надлишковий код, програмування на мові С.

DATA EXCHANGE IN MICROCONTROLLERS WITH EXTERNAL ENERGY INDEPENDENT MEMORY ON TWO-WIRE SEQUENTIAL INTERFACE WITH INTEGRITY CHECK USING THE CRC-8/MAXIM CYCLIC REDUNDANCY CHECK ALGORITHM

Losikhin D.A.

Ukrainian State University of Chemical Technology, Dnipro, Ukraine

The article is devoted to the organization of serial communication between the microcontroller and an external programmable read-only memory device with data integrity control. Obtaining reliable information on information-measuring channels in automation systems is largely provided by intelligent sensors, the main hardware elements of which are primary measuring transducers, a microcontroller, non-volatile memory and their interface components. At the software level, the computing capabilities of a microcontroller make it possible to implement algorithms for the primary processing of information, diagnostic self-monitoring and algorithms for adapting to changing external influences in smart sensors. In this case, calibration data for correction of the measurement error, parameters of the self-healing algorithm when a single defect occurs, parameters of self-learning algorithms are located in the external non-volatile memory. However, the risk of obtaining an unreliable measurement result also depends on the state of hardware of a smart sensor itself, in particular, on the endurance of non-volatile memory, and on data transmission errors on digital interface channels, in particular, on a serial communication interface between microcontroller and non-volatile memory. To increase the reliability of data transmission and storage in information carriers, redundant coding is applied using schemes and algorithms for calculating cyclic redundant code (CRC). Support for CRC calculations at the hardware level is not provided by all non-volatile memory microcircuits, and microcontrollers, unfortunately, do not have such hardware error control encoders in serial data stream. The solution to the problem at the software level is examined using an example of data exchange between the ATmega328p microcontroller (Microchip Technology Inc.) and the external non-volatile memory AT24C04C/AT24C08C (Microchip Technology Inc.) via the I2C two-wire serial interface with integrity checking using the 8-bit algorithm CRC-8 cyclic redundancy code (Maxim Integrated). A corresponding library of functions in C language and a test program have been developed. The develoded code is verified on real devices.

Keywords: measuring system, smart sensor, microcontroller, external electrically erasable programmable read-only memory, serial synchronous asymmetric communication bus of integrated circuits, data integrity control, cyclic redundancy code, C programming.

REFERENCES

1. Lapin A.A. *Interfeiysy. Vybor i realizatsiya* [Interfaces. Selection and implementation]. Moskva: Tekhnosphera, 2005, 168 p. (in Russian).

2. Understanding and Using Cyclic Redundancy Checks with Maxim 1-Wire and iButton Products (URL: https://pdfserv.maximintegrated.com/en/an/AN27.pdf).

3. Klyuchev A.O., Kovyazina D.R., Petrov E.V., Platunov A.E. *Interfeysy periferiynykh ustroystv*. [Peripheral Interfaces]. SPb.: SPbGU ITMO, 2010, 290 p. (in Russian).

4. ATmega328P. DATASHEET (URL: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf).

5. AT24C04C and AT24C08C. DATASHEET (URL: http://ww1.microchip.com/downloads/en/devicedoc/atmel-8787-seeprom-at24c04c-08c-datasheet.pdf).

6. Chopra R. C Programming: A Self-Teaching Introduction. New Delhi: New Age International Publishers, 2018.

7. Avdeyev V.A. *Periferiynyye ustroystva: interfeysy, skhemotekhnika, programmirovaniye* [Peripherals: interfaces, circuitry, programming]. Moskva: DMK Press, 2009, 848 p. (in Russian).