Современные 32-разрядные ARM-микроконтроллеры серии STM32: блок вычисления кода CRC

Олег Вальпа (г. Миасс, Челябинская обл.)

В статье приведено описание блока вычисления кода CRC 32-разрядных ARM-микроконтроллеров серии STM32 от компании STMicroelectronics. Рассмотрен состав и назначение регистров блока, а также даны примеры программ для работы с этим блоком.

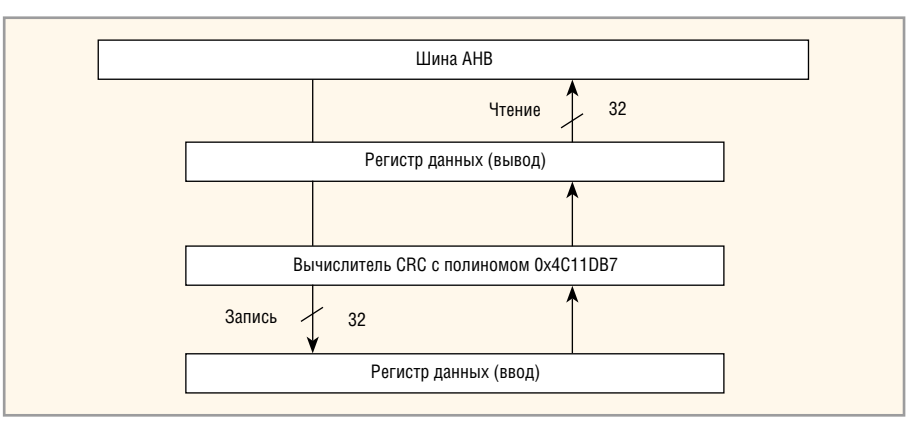
Введение

В большинстве протоколов передачи данных для обеспечения достоверности их передачи обычно используется контрольная сумма произвольного блока данных. Для повышения качества проверки данных вместо контрольной суммы можно применять специальный код, так называемый CRC (Cyclic Redundancy Code – циклический избыточный код). Обычно используют коды с разрядностью 8, 16 или 32 бит. Для вычисления такого кода применяются исходные полиномы определённого вида.

В отличие от простой контрольной суммы код CRC позволяет обнаружить искажение даже в одном бите большого массива данных. Если обнаружено несовпадение кода CRC, вычисленного на приёмной стороне для блока принятых данных, и принятого кода CRC – данные считаются недостоверными и отбраковываются. В этом случае приёмник данных посылает сообщение об ошибке кода CRC и осуществляется повторная передача данных. Таким образом, производится защита данных от искажения во время обмена между устройствами.

В принципе код CRC можно вычислять с помощью программы. Однако на такое вычисление требуется определённое время. И если обмен данными происходит регулярно, то процессору потребуется отводить немало времени на подсчёт кодов CRC для каждой передачи или приёма данных.

Для снижения затрат времени на эти операции 32-разрядные ARM-микроконтроллеры серии STM32 [1] были оснащены аппаратным блоком вычисления контрольной суммы данных в виде кода CRC.



Блок-схема работы блока вычисления CRC

Карта регистров блока вычисления CRC

Сдвиг	Регистр	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	CRC_DR Исх. знач.	Регистр данных 0xFFFF FFFF										
0x04	CRC_IDR Исх. знач.	Резерв			Регистр независимых данных 0x00							
0x08	CRC_CR Исх. знач.	Резерв										Сброс 0

Описание блока

На рисунке приведена диаграмма работы блока вычисления CRC микроконтроллера STM32.

Вычисление кода CRC в блоке производится с помощью 32-разрядного регистра ввода и вывода и вычислителя CRC с полиномом. Подсчёт кода производится за 4 тактовых цикла шины AHB с тактовой частотой HCLK.

Блок вычисления микроконтроллера STM32 использует для получения 32-битного слова кода CRC полином вида: X^32 + X^26 + X^23 + X^22 + X^16 + X^12 + + X^11 + X^10 + X^8 + X^7 + X^5 + X^4 + X^2 + + X + 1. Этот полином можно представить в виде шестнадцатеричного числа: 0x04C11DB7. Каждый разряд этого числа определяет наличие или отсутствие члена полинома X в соответствующей степени.

Блок вычисления использует 32-разрядный регистр данных как входной регистр для записи очередных данных для вычислителя кода CRC, а также хранит в нём результат предыдущего вычисления CRC, используемый при чтении этого регистра. Каждая операция записи в регистр данных создаёт комбинацию предыдущего значения CRC и его нового значения. Вычисление кода CRC выполняется не побайтно, а сразу над целым 32-разрядным словом.

Вычисление кода CRC обычно используется для проверки корректности передачи и приёма данных. Кроме того, в соответствии со стандартом EN/IEC 60335-1 код CRC может использоваться для проверки целостности содержимого флэш-памяти. Код CRC программы вычисляется в контроллере во время его работы, а затем сравнивается с контрольной суммой, которая была подсчитана во время её предварительной компоновки и была сохранена в определённой области памяти. При их несовпадении формируется соответствующее прерывание, и выполнение программы прекращается.

Регистры блока

Блок вычисления CRC включает в свой состав следующие регистры:

- регистр данных CRC_DR;
- регистр независимых данных CRC_IDR;

- регистр управления CRC_CR.

В таблице приведена карта регистров блока вычисления CRC, а также и их значения после сброса.

Регистр данных CRC_DR используется как входной регистр при записи новых данных в вычислитель CRC. При чтении он содержит предыдущий результат вычисления CRC.

Регистр независимых данных CRC_IDR используется как 8-разрядный регистр общего назначения. Он может использоваться для временного хранения одного байта. Данные этого регистра сохраняются при сбросе CRC, вызванного битом регистра CRC_CR.

Регистр управления CRC_CR содержит всего один бит, используемый как бит сброса. Он сбрасывает блок вычисления CRC и устанавливает регистр данных в исходное значение 0xFFFF FFFF. Этот бит программно можно только устанавливать. Сбрасывается он автоматически, аппаратно.

Более подробную информацию о блоке вычисления кода CRC можно получить в оригинальном источнике [2].

Программирование

Библиотека CMSIS предоставляет целый набор функций для работы с блоком CRC. Функции этой библиотеки и подробные комментарии к ним представлены в листинге 1.

Для инициализации блока вычисления кода CRC и работы с ним необходимо выполнить следующие действия:

- разрешить тактирование блока;
- произвести сброс данных блока;
- записать массив данных в блок;
- считать вычисленное значение кода CRC из блока.

Рассмотрим конкретный пример программы, приведённый в листинге 2, которая позволяет вычислить код CRC для массива данных.

В результате работы программы переменная data_crc будет иметь значение 0x4B6B373E. Это и есть код CRC, который можно передавать вместе с массивом данных по каналу связи.

Таким образом, с помощью блока вычисления кода CRC обеспечивается надёжный обмен данными с внешним устройством без увеличения нагрузки на процессор микроконтроллера STM32.

Литература

1. https://www.st.com.
2. http://www.st.com/web/en/resource/technical/document/reference_manual/CD00246267.pdf.

Листинг 1

```
// Функция вычисления кода CRC для массива 32-битовых слов
uint32_t CRC_CalcBlockCRC (uint32_t pBuffer[], uint32_t BufferLength)
// Вход:
// pBuffer - указатель на массив данных для подсчёта контрольной суммы
// BufferLength - размер массива данных
// Выход: Контрольная сумма для массива слов
// Функция вычисления кода CRC для одного 32-разрядного слова
uint32_t CRC_CalcCRC (uint32_t Data)
// Вход: Data - 32-разрядное слово, для которого требуется вычислить
код CRC
// Выход: Контрольная сумма для одного слова
// Функция для чтения текущего значения CRC
uint32_t CRC_GetCRC (void)
// Вход: Параметров нет
// Выход: Текущее значение регистра CRC_DR
// Функция для записи 8-разрядных данных в регистр данных CRC_IDR
void CRC_SetIDRegister (uint8_t IDValue)
// Вход: IDValue - данные для записи
// Выход: Значения не возвращает
// Функция для чтения 8-разрядных данных из регистра данных CRC_IDR
uint8_t CRC_GetIDRegister (void)
// Вход: Параметров нет
// Выход: Текущее значение регистра CRC_IDR
// Функция для сброса регистра данных CRC_DR
void CRC_ResetDR (void)
// Вход: Параметров нет
// Выход: Значения не возвращает
```

Листинг 2

```
#include "stm32f10x.h"
#include "stm32f10x_rcc.h"
//Функция для аппаратного вычисления кода CRC
//Вход: Указатель на массив данных и размер этого массива
//Выход: Код CRC для входного массива данных
uint32_t crc_calc(uint32_t * buffer[], uint32_t buff_len) {
uint32_t i;
CRC->CR |= CRC_CR_RESET; // Сбросить данные в блоке
// Записать данные из буфера в регистр данных
for(i = 0; i < buff_len; i++)
{
CRC->DR = buffer[i];
}
return (CRC->DR); // Читать код CRC
}
// Главный модуль программы
int main(void)
{
// Объявить и заполнить массив данных
uint32_t data[]={0x5B27E2BE,0xFF2711BC,0xABA7EE00};
// Разрешить тактирование блока вычисления кода CRC
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_CRC, ENABLE);
uint32_t data_crc=crc_calc(&data,3); //Вычисляем CRC для массива данных
// Бесконечный цикл для других команд программы
while(1)
 {
 // Здесь можно разместить другие команды программы
 }
}
```