ОПТИМИЗАЦИЯ КОДА ПРИ ПРОГРАММИРОВАНИИ AVR МИКРОКОНТРОЛЛЕРОВ НА ЯЗЫКЕ С

Гурков С.А.
Научный руководитель – к.т.н, доц. Свид И.В.
Харьковский национальный университет радиоэлектроники
(61166, Харьков, пр. Науки,14, каф. Радиотехнологий информацион
но-коммуникационных систем, тел. +38(057)7021-444)
e-mail: serhii.hurkov@nure.ua

The report describes some of the ways to optimize the code at prograг ming AVR microcontrollers with using high-level language C.

Программирование микроконтроллеров, как правило,
Ассемблер и Первый машинно-ориентированный
язык низкого уровня. Второй – язык высокого уровня общего

Основными недостатками Ассемблера являются жесткая
да к конкретному семейству контроллеров, и перенос кода
контроллера на другой. При
тем не менее, фатальные ошибки.

К преимуществам С относится сокращенное время написания
граммы, более простой поиск ошибок, а также легкость повторного пр
нения. К недостаткам можно отнести – более громоздкий результирующий код и сниженное быстродействие. Основной недостаток С, как языка программирования для микроконтроллеров, перед Ассемблером – более низкое быстродействие.

Основные приемы позволяющие увеличить эффективность использования языка программирования С перечислены ниже.

Использование локальных переменных. Если нет необходимости использовать переменную в других частях кода, то выгоднее сделать её локальной. Этот прием положительно скажется как на быстродействии, так и на размере программы.

Выбор наиболее приемлемых типов данных. В том числе, знаковые типы.

Объявление нелокальных переменных как статические, если они упоминаются в пределах лишь одной функции.

Использование простых условий для организации вечных циклов.For(;;), while(true), do{} while(true).Не стоит забывать, что 0 – ложь, значение отличное от нуля – истина.

Использование прямого доступа к памяти ввода/вывода, вместо указателей.

Объединениенескольких функций в один модуль (т.е. в один файл) для увеличения возможности повторного использования кода. Этот прием не слишком эффективен для снижения размера программы, и увеличения быстродействия, но он может существенно упростить читаемость кода и

поиск ошибок.

Использование минимально возможной модели памяти.

Уточнение точных значений размеров программного стека и стека возврата.

Хранение всех констант и переменных во флэш-памяти с помощью зарезервированных слов.

Использование типа void для функций, которые не возвращают значения, т.к. под такие функции не будет выделяться память.

Отказ от использования вещественных типов данных, таких как float. Во первых, использование вещественных типов приводит к быстрому заполнению памяти программ. Во вторых вычисления выполняются очень медленно. Но, если дробные переменные необходимы, то решением может стать использование чисел с фиксированной запятой. Число с фиксированной запятой, обычно состоящее из байта или слова из двух байт, содержит целую часть, которая находится в старших битах,и дробную, находящуюся в младших битах. Математические операции с такими числами ничем не отличаются от математических операций с простыми целыми числами. Код получается компактный и быстрый. Для использования данного приема необходимо выбрать разрядность, слово или байт, и положение запятой. Например, для элементарного вольтметра на базе встроенного в atmega АЦП, который будет работать в диапазоне напряжений от 0 до 2,5В достаточно 1-го байта. Целая часть переменной будет храниться в 7-ом и 8-ом битах. Точность такого вольтметра достигает 0,016В. При подаче напряжения в 1,7 В, байт примет значение 01101101, что соответствует напряжению (64+32+8+4+1)/64 = 1,703 В.

Приведенные выше приемы позволяют повысить эффективность использования ресурсов микроконтроллеров AVR.

В языке С гармонично сочетаются возможности программирования низкого уровня со свойствами языка высокого уровня. Возможность низкоуровневого программирования позволяет легко оперировать непосредственно аппаратными средствами, а свойства языка высокого уровня позволяют создавать легко читаемый и модифицируемый программный код. Кроме того, практически все компиляторы С имеют возможность использовать ассемблерные вставки для написания критичных по времени выполнения и занимаемым ресурсам участков программы.

Список литературы

1. AVR035: Efficient C Coding for AVR - Atmel corporation, 2003.

2. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров - МК-Пресс, 2006.