УДК 681.518

Н.В. Смирнова, доц., канд. техн. наук, В.В. Смирнов, доц., канд.техн. наук
Кировоградский национальный технический университет

Метод организации параллельного выполнения задач в микроконтроллерах с малым объемом памяти программ

Приведено описание метода организации параллельного выполнения задач в микроконтроллерах с малым объемом памяти программ. Ресурсы микроконтроллеров начальных серий не позволяют использовать полноценную многозадачную операционную систему реального времени, тем самым значительно ограничивая возможности разрабатываемой программы. Реализация многозадачности на основе прерываний таймера, статических переменных, учета приоритета потоков и флагов состояний позволяет реализовывать управляющие программы на микроконтроллерах с ограниченными ресурсами.
потоки выполнения, параллелизм задач, микроконтроллер, программа, прерывание, RTOS

Н.В. Смірнова, доц., канд. техн. наук В.В. Смірнов, доц., канд. техн. наук
Кіровоградський національний технічний університет
Метод організації паралельного виконання задач у мікроконтролерах з малим об'ємом пам'яті програм

Наведено опис методу організації паралельного виконання завдань у мікроконтролерах з малим об'ємом пам'яті програм. Ресурси мікроконтролерів початкових серій не дозволяють використовувати повноцінну многозадачную операційну систему реального часу, тим самим значно обмежуючи можливості розроблюваної програми. Реалізація багатозадачності на основі переривань таймера, статичних змінних, врахування пріоритету потоків і прапорів станів дозволяє реалізовувати керуючі програми на мікроконтролерах з обмеженими ресурсами.
потоки виконання, паралелізм завдань, мікроконтролер, програма, переривання, RTOS

Постановка проблемы. В настоящее время существует большое количество разновидностей микроконтроллеров для решения большого круга задач управления объектом. Микроконтроллеры отличаются один от другого как базовой архитектурой, объемом памяти программ и данных, так и стоимостью.

Целесообразность выбора конкретного типа микроконтроллера определяется целевой задачей. В ряде случаев, возможности микроконтроллера являются избыточными для решения конкретной задачи при выполнении однопоточной (последовательной) программы. В то же время, для параллельного выполнения нескольких задач или выполнения нескольких потоков в рамках одной задачи, возможностей микроконтроллера недостаточно.

В первую очередь, это обусловлено небольшим объемом памяти программ и оперативной памяти микроконтроллера. Обычное решение такой проблемы достаточно простое: выбирается более мощный микроконтроллер с ресурсами, достаточными для функционирования операционной системы реального времени. Такая операционная система, например FreeRTOS, легко обеспечивает параллелизм выполнения нескольких программных потоков для выполнения нескольких задач.

Анализ исследований и публикаций. Среди существующих многозадачных операционных систем реального времени только операционная система FreeRTOS поддерживает наибольшее количество архитектур микропроцессоров и

микроконтроллеров, включая PIC-контроллеры, начиная с серии 18 [1]. Более младшие серии PIC-контроллеров, такие, как 10/12 и 16 не поддерживаются ни одной RTOS.

Постановка задачи. Необходимость создания облегченной версии многозадачной операционной системы вытекает из тех преимуществ, которые реализует механизм многозадачности.

Функции циклов опроса клавиатуры и различных датчиков могут быть реализованы в виде отдельных задач, которые могут выполняться в отдельных потоках, не мешая работе основного процесса.

Например, в однопоточной программе ожидание ввода команды пользователя (оператора) с клавиатуры останавливает выполнение основной программы до момента нажатия клавиши. В многопоточной программе задача опроса клавиатуры и датчиков может осуществляться в фоновом режиме параллельно выполнению основной задачи.

На самом деле, «чистый» параллелизм не может быть реализован на одном процессоре, тем не менее, выполнение нескольких задач в режиме разделения времени не создает каких-либо препятствий для реализации механизма многозадачности.

Цель работы - создание метода организации параллельного выполнения нескольких задач в параллельных потоках для микроконтроллеров с ограниченными ресурсами, который позволит снять ограничения, присущие однопоточным программам и позволит реализовать преимущества многозадачности в контроллерах с малым объемом памяти программ.

Основная часть. Основой многозадачной операционной системы является планировщик задач [2]. В функцию планировщика входит назначение приоритетов задачам, планирование выполнения задач, запуск задач на выполнение, сохранение и переключение контекста выполнения задач и ряд других системных функций.

В микроконтроллерных системах термины «поток» и «задача» можно считать синонимами, поскольку задачей является отдельная функция или блок функций программы, имеющей одну точку входа – функцию `main()`.

Жизненный цикл потока для выполнения задачи представлен на рис. 1.

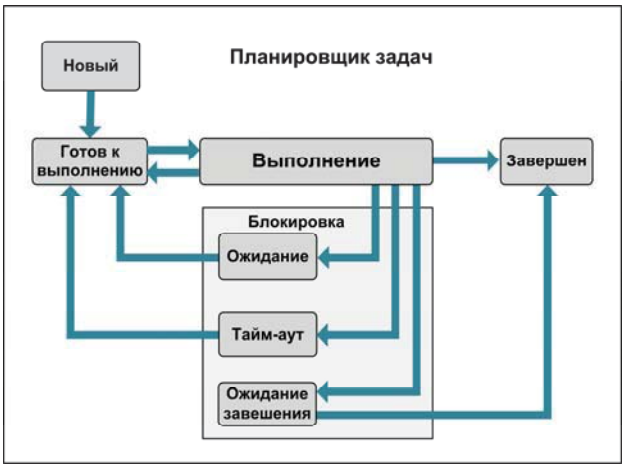

Рисунок 1 – Жизненный цикл потока в многозадачной операционной системе

Поток может находиться в состояниях: готовности к выполнению, выполнения, блокировки и завершения. Состояниями потока руководит Планировщик задач (Sheduler). Объем программного кода полноценной многозадачной операционной системы (более 8 Кб) значительно превышает объем памяти программ микроконтроллеров начальных серий (1-4 Кб) [3, 4].

Реализация полновесного планировщика задач фактически является реализацией

ядра многозадачной операционной системы, что не дает никаких преимуществ в плане минимизации объема программного кода. Поэтому механизм многозадачности реализован в виде очереди, в которой находятся потоки, ожидая запуска на выполнение. В отличие от полноценных многозадачных операционных систем, в которых невозможно предугадать, какой поток будет запущен в следующий момент времени, в упрощенном механизме реализации многозадачности запуск каждого потока осуществляется в строгой очередности. Вновь созданный поток помещается в очередь типа FIFO и ожидает запуска (рис.2).

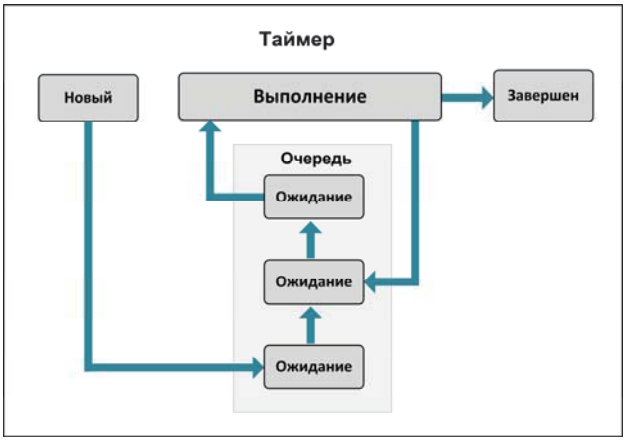

Рисунок 2 – Запуск потоков в очереди FIFO

Таймер переключения потоков вызывает прерывания с периодом 1 мс. В обработчике прерывания таймера осуществляется запуск потоков на выполнение в соответствии с алгоритмом, в котором проверяется флаг активности потока, значение которого соответствует уровню приоритета потока (рис 3.).

Рисунок 3 – Алгоритм работы планировщика задач на основе прерываний таймера

Важным аспектом функционирования любой многозадачной операционной системы является сохранения контекста неактивных потоков в оперативной памяти. В данном случае контекст выполнения потоков не сохраняется по следующим причинам:

- На сохранение контекста потока недостаточно оперативной памяти.
- В этом нет необходимости, поскольку компилятор при компиляции распределяет оперативную память для контекста выполнения потоков и потери значений флагов состояния задач и данных не происходит.

При этом необходимо выполнить условие: при разработке программы, для потоков, выполнение которых связано с ожиданием выполнения и завершения длительных операций, необходимо объявлять либо глобальные, либо статические переменные.

Для потоков, которые выполняются за время, меньшее, чем интервал работы таймера (1мс), можно объявлять локальные переменные.

Поскольку запуск и выполнение потоков является строго синхронным процессом, то гарантируется, что поток, цикл выполнения которого меньше периода работы таймера не будет прерван.

При такой организации многозадачности гарантируется минимальное время реакции управляющей программы на внешние или внутренние прерывания.

Функциональная схема реализация механизма многозадачности в управляющей программе отладочного модуля представлена на рис. 4.

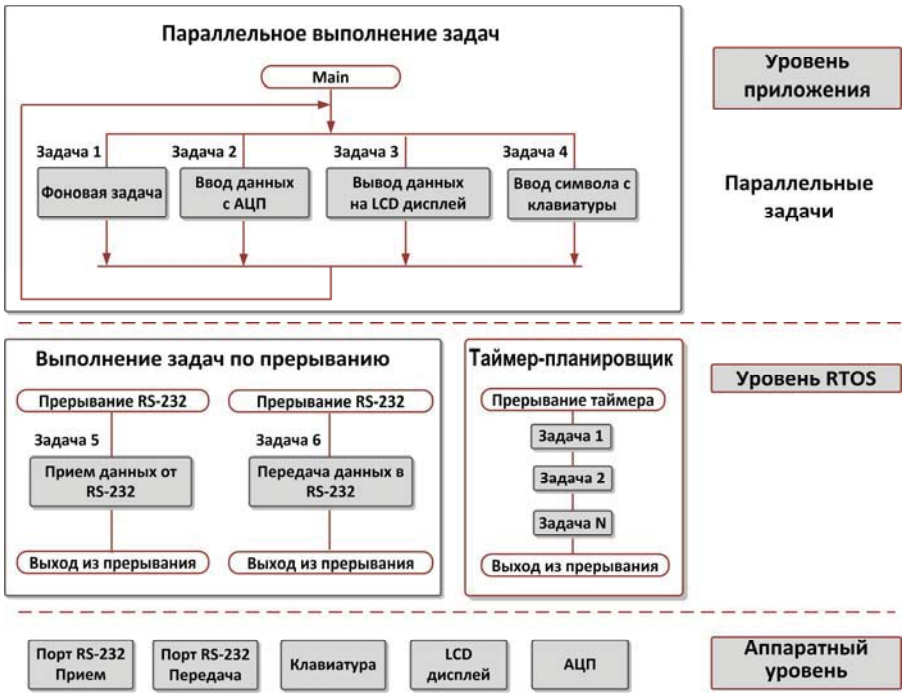

Рисунок 4 – Функциональная схема управляющей программы отладочного модуля

Отладочный модуль реализован на микроконтроллере PIC16F873A с объемом памяти программ 4 Кб.

Параллельное выполнение потоков: фоновой задачи, ввода данных с АЦП, вывода данных на LCD дисплей и ввода символов с клавиатуры осуществлялось без каких-либо сбоев, потерь данных или зависаний программы. Обмен данными по интерфейсу RS-232 осуществлялся в режиме обработки аппаратных прерываний.

Приоритет потока назначается при его создании. Величина минимального приоритета MIN_PRIORITY = 1, максимального приоритета MAX_PRIORITY = 10. По

умолчанию, при создании потока ему присваивается значение приоритета NORM_PRIORITY = 5.

Приоритет потока определяет количество периодов таймера, в течении которых будет выполняться запущенный поток. То есть, если приоритет одного потока равен 3, а второго потока – 6, то это означает, что первому потоку отпущен лимит времени выполнения 3 мс (3 периода таймера), а второму потоку – 6 мс (6 периодов таймера).

При возникновении аппаратного или программного прерывания обработка прерывания таймера переключения потоков блокируется до момента завершения обработки прерывания. Таким образом, порядок запуска потоков на выполнение не нарушается.

Результаты разработки «облегченного» механизма реализации многозадачности дают основания для рекомендации его использования в микроконтроллерах с небольшими ресурсами при создании управляющих программ небольшого объема.

Выводы. Разработанный метод организации параллельного выполнения нескольких задач в параллельных потоках не является конкурентом полноценных многозадачных операционных систем реального времени для микроконтроллеров.

Метод разработан для применения параллелизма выполнения нескольких задач в микроконтроллерах, ресурсы которых не позволяют использовать полноценную операционную систему.

Реализация метода в нескольких проектах показала его эффективность для задач, решение которых в однопоточных программах очень затруднено.

Таким образом, применение разработанного метода организации параллельного выполнения задач позволяет использовать микроконтроллеры с небольшим объемом памяти программ, облегчить процедуру разработки программ управления для них и создать более качественную программную систему при прочих равных условиях.

Список литературы

1. FreeRTOS Quality RTOS & Embedded Software [Электронный ресурс]: The Market Leading, De-facto Standard and Cross Platform Real Time Operating System (RTOS) / FreeRTOS. - Режим доступу: http://freertos.org.
2. Э. Таненбаум Операционные системы. Разработка и реализация / Э. Таненбаум, А. Вудхалл. – СПб.: Питер, 2007. – 704 с.
3. PIC12F629/675 Data Sheet 8-Pin FLASH-Based 8-Bit CMOS Microcontrollers. - Microchip.com. – 110 p.
4. PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers.- Microchip.com. – 232 p.

Nataliya Smirnova, Vladimir Smirnov
Kirovograd National Technical University

The tasks parallel execution in the microcontroller with low program memory programs organization method

The method parallel tasks in microcontrollers with low memory programs implementation description. Resources microcontrollers initial series does not allow the use of a full-fledged multi-tasking real time operating system, thereby significantly limiting the ability to develop the program. The implementation of multitasking on the basis of timer interrupts, static variables, taking into account the priority of threads and flags of states allows to implement control programs for microcontrollers with limited resources.

Multitasking is implemented as a FIFO queue within which streams are waiting to start execution. Unlike full-fledged multi-tasking operating system, which is impossible to predict which thread will be launched in the next moment, in a simplified mechanism for multitasking start of each stream is carried out in order of priority.

execution threads, task parallelism, a microcontroller, a program interruption, RTOS

Получено 20.04.15