Применение SWITCH-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Часть 5

Владимир ТАТАРЧЕВСКИЙ
arktur04@mail.ru

В предыдущих статьях цикла [1, 2] рассматривался механизм реализации таймеров в SWITCH-программах. В данной статье будет продолжено обсуждение реализации таймеров.

В статье [1] при рассмотрении механизма работы таймеров виртуальные таймеры в SWITCH-программе были условно разделены на два типа: локальные и глобальные. При этом локальным таймером был назван таймер, отсчитывающий время с момента входа в состояние[1] конечного автомата, который используется для определения условий выхода из данного состояния по истечении определенного интервала времени. Также было отмечено, что глобальный таймер, область действия которого охватывает несколько состояний, обладает гораздо большей гибкостью (рис. 1). Итак, рассмотрим пример применения глобального таймера и его программную реализацию.

Глобальные таймеры

Глобальным таймером будем называть виртуальный таймер, значение которого инициализируется в одном состоянии автомата, а применяется в качестве условия перехода — в другом. С его помощью можно придать временной детерминизм процессу, который управляется группой состояний автомата.

Для того чтобы понять, зачем нужны глобальные таймеры, рассмотрим в качестве примера систему управления установкой подачи воды в трубопровод. Функциональная схема установки подачи воды в трубопровод приведена на рис. 2. Вода поступает в бак через клапан налива, а затем поступает из бака в технологический трубопровод через клапан подачи.

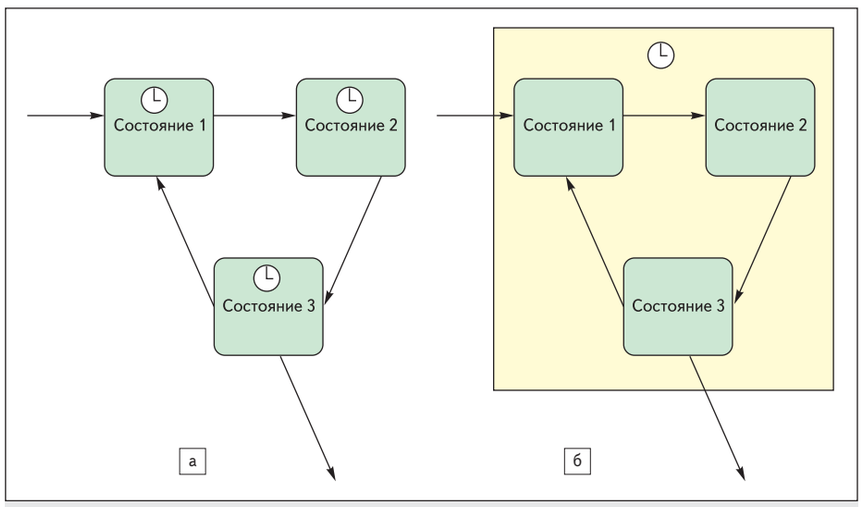

Рис. 1. Локальные и глобальные таймеры:
а) локальный таймер запускается в каждом состоянии при входе в него конечного автомата;
б) глобальный таймер является общим для группы состояний автомата

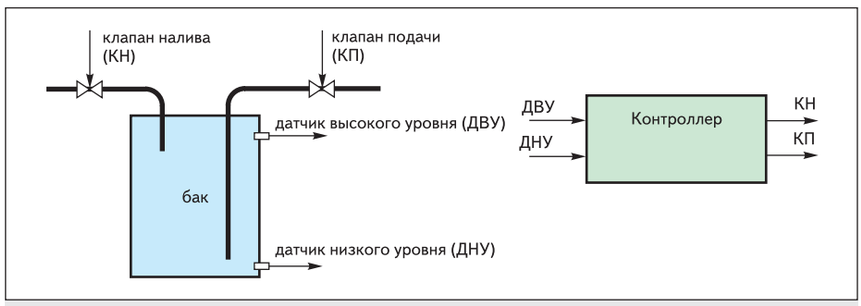

Рис. 2. Функциональная схема установки подачи воды в трубопровод

Система должна подавать в трубопровод воду в количестве (предположим) десяти объемов бака за один рабочий цикл. С целью экономии оборудования обойдемся без расходомера, а будем вычислять объем воды исходя из времени заполнения бака. При этом расход воды в единицу времени заранее неизвестен, но предполагается, что он существенно

1 В каждое состояние автомата, разумеется (*прим. автора*).

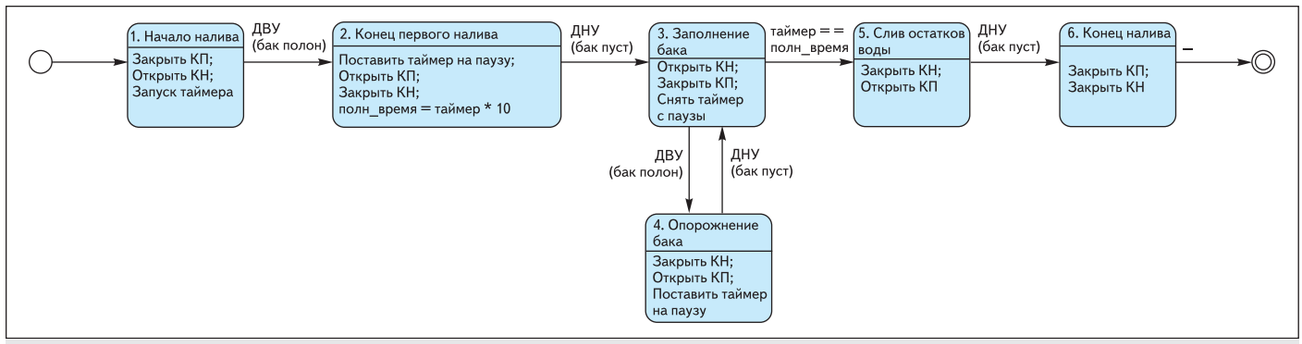

Рис. 3. Конечный автомат системы управления установкой подачи воды

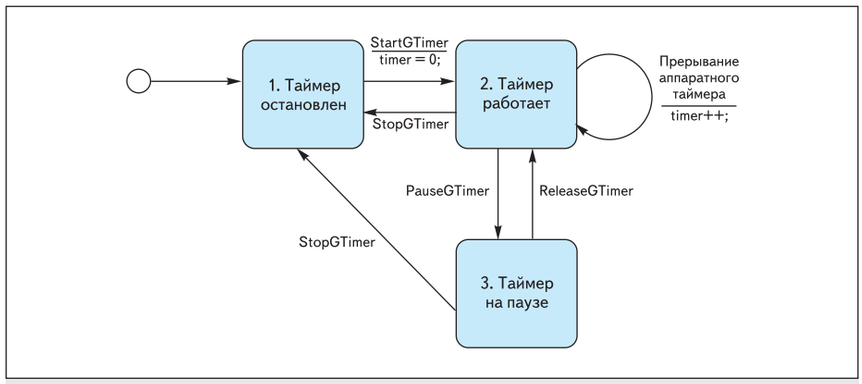

Рис. 4. Конечный автомат, описывающий глобальный таймер

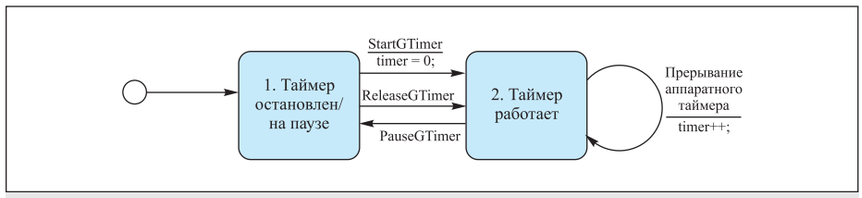

Рис. 5. Упрощенный конечный автомат глобального таймера

не изменяется за время одного рабочего цикла. Скорость заполнения бака легко измерить по времени его заполнения (от срабатывания датчика нижнего уровня до срабатывания датчика верхнего уровня) при отсутствии отбора воды из бака (клапан подачи закрыт). Пусть при этом скорость набора воды в бак выше скорости отбора воды из бака. Поэтому нельзя просто открыть клапан налива и отсчитать определенное время, так как бак переполнится. Следовательно, необходимо применить более гибкий подход.

Изобразим алгоритм работы установки в виде конечного автомата с глобальным таймером (рис. 3).

Цикл системы начинается с открывания клапана налива. При этом клапан подачи закрыт. Одновременно с началом рабочего цикла начинается отсчет времени таймером (состояние 1 «Начало налива»). Поступление воды продолжается до достижения верхнего уровня бака (состояние 2 «Конец первого налива»). После этого закрывается клапан налива, измеряется время заполнения бака, вычисляется полное время налива требуемого количества воды, открывается клапан подачи в трубопровод. После осушения бака автомат переходит в состояние «Продолжение налива». По мере заполнения бака и его опустошения автомат переключается между состояниями 3 «Опорожнение бака» и 4 «Заполнение бака». При этом отсчет времени происходит только при открытом клапане налива. По прохождении полного времени налива автомат переходит в состояние 5 «Слив остатка воды», а затем завершает работу (состояние 6 «Конец налива»).

На приведенном примере хорошо видны отличия глобального таймера от локального. Первое отличие, уже обсуждавшееся выше, заключается в том, что запуск таймера происходит в одном состоянии автомата, а его значение служит условием перехода в другое состояние (или несколькие состояния). Второе отличие состоит в том, что отсчет времени глобальным таймером можно приостановить («поставить на паузу»)[2], а потом снова запустить. Благодаря этому мы можем контролировать время протекания не только непрерывных процессов, но и таких процессов, выполнение которых периодически приостанавливается (как набор воды в приведенном примере). Третья особенность глобального таймера состоит в том, что мы можем в любой момент времени прочитать его текущее значение и использовать в вычислениях (для локального таймера мы тоже можем прочитать его текущее значение, но так как он «существует» только в пределах одного состояния, это не так интересно с практической точки зрения).

Итак, поняв, что такое глобальный таймер и для чего он нужен, опишем его функционирование более формально. Для обеспечения работы с глобальными таймерами используем следующие функции:

```
void StartGTimer(unsigned int GTimerID); //Запуск таймера
void StopGTimer(unsigned int GTimerID); //Останов таймера
void PauseGTimer(unsigned int GTimerID); //Приостановка работы таймера
void ReleaseGTimer(unsigned int GTimerID); //Продолжение работы таймера
unsigned int GetGTimer(unsigned int GTimerID); //Получение текущего значения таймера
```

Теперь мы легко сможем описать поведение глобального таймера в виде конечного автомата (рис. 4).

Отметим, что на практике конечный автомат, приведенный на рис. 3, можно упростить, объединив состояния 1 и 3 и избавившись от функции StopGTimer (так как она становится эквивалентной функции PauseGTimer). Упрощенный таким образом конечный автомат изображен на рис. 5.

Перейдем к программной реализации механизма глобальных таймеров.

2 Может быть, выражение «поставить таймер на паузу» несколько громоздко и не слишком удачно, но позволяет более четко разграничить состояние полного останова таймера, в котором он сброшен «в ноль», и состояние временной приостановки его работы, в котором он сохраняет текущее значение счетчика времени. Здесь можно провести аналогию с магнитофонной кассетой, которую можно перемотать к началу (перевести таймер в состояние останова), или нажать кнопку «пауза», а затем, через какое-то время, продолжить воспроизведение (*прим. автора*).

Программная реализация механизма глобальных таймеров

```
//-----------------
// Модуль timers.h
//-----------------
#ifndef TIMERS_h
#define TIMERS_h

.
.
.

#define MAX_GTIMERS 16 //Максимальное к-во глобальных таймеров в системе

//Идентификаторы глобальных таймеров
#define GTIMER1              0
#define GTIMER2              1
.
.
.
#define GTIMER15             11

//Функции работы с глобальными таймерами
void InitGTimers(void); //Инициализация глобальных таймеров
void StartGTimer(unsigned int GTimerID); //Запуск таймера
void StopGTimer(unsigned int GTimerID); //Останов таймера
void PauseGTimer(unsigned int GTimerID); //Приостановка работы таймера
void ReleaseGTimer(unsigned int GTimerID); //Продолжение работы таймера
unsigned int GetGTimer(unsigned int GTimerID); //Получение текущего значения таймера
.
.
.
#endif

//-----------------
// Модуль timers.c
//-----------------
#include «timers.h»

unsigned int GTimers[MAX_GTIMERS]; //В массиве хранятся текущие значения глобальных таймеров

//Состояния таймера
#define TIMER_STOPPED 0 //Таймер остановлен
#define TIMER_RUNNING 1 //Таймер работает
#define TIMER_PAUSED  2 //Таймер на паузе

char GTStates[MAX_GTIMERS]; //В массиве хранятся текущие состояния глобальных таймеров

void InitGTimers(void) //Инициализация глобальных таймеров
{
   char i;
   for(i = 0; i < MAX_GTIMERS; i++)
       GTStates[i] = TIMER_STOPPED;
}
void StartGTimer(unsigned int GTimerID) //Запуск таймера
{
   if(GTStates[GTimerID] == TIMER_STOPPED)
   {
       GTimers[GTimerID] = 0;
       GTStates[GTimerID] = TIMER_RUNNING;
   }
}
void StopGTimer(unsigned int GTimerID) //Останов таймера
{
   GTStates[GTimerID] = TIMER_STOPPED;
}
void PauseGTimer(unsigned int GTimerID) //Приостановка работы таймера
{
   if(GTStates[GTimerID] == TIMER_RUNNING)
       GTStates[GTimerID] = TIMER_PAUSED;
}
void ReleaseGTimer(unsigned int GTimerID) //Продолжение работы таймера
{
   if(GTStates[GTimerID] == TIMER_PAUSED)
       GTStates[GTimerID] = TIMER_RUNNING;
}
unsigned int GetGTimer(unsigned int GTimerID) //Получение текущего значения таймера
{
   return GTimers[GTimerID];
}

void ProcessTimers(void) //обработчик прерывания таймера/счетчика
{
   char i;
…
   for(i = 0; i < MAX_GTIMERS; i++)
       if(GTStates[i] == TIMER_RUNNING)
           GTimers[i]++;
…
}
```

Небольшое отступление. Как видно из приведенного выше листинга, программа, которая реализует управление глобальными таймерами, построена по графу конечного автомата, показанного на рис. 4, предназначена для использования в SWITCH-программах, однако сама по себе построена не по SWITCH-технологии. Это вызвано тем, что переходы между состояниями таймера происходят под воздействием вызовов функций и наиболее удобной в данном случае является реализация этих переходов непосредственно в функциях управления таймером, а не в едином операторе switch. Таким образом, мы несколько пожертвовали «чистотой концепции» ради простоты реализации.

Сейчас мы легко сможем построить программу, реализующую конечный автомат, изображенный на рис. 3.

```
char state; //переменная состояния автомата
char _state; //предыдущее состояние автомата
char entry; //флаг = 1 при входе автомата в новое состояние

unsigned int full_time;//полное время налива
void InitFSM(void)
{
   state = _state = 0;
   entry = 0;
}
void ProcessFSM(void)
{
   if(state != _state) entry = 1; else entry = 0;
   switch(state)
   {
      case 0: //Неактивное состояние
         if(GetMessage(MSG_ACTIVARE)) state = 1;
         break;
      case 1: //Начало налива
         if(entry) //при входе в состояние
            {
               StartGTimer(TIMER1);//Запускаем таймер
               SetOut(KP, off);//Закрываем клапан подачи
               SetOut(KN, on);//Открываем клапан налива
            };
         if(GetInput(DVU))//Если бак заполнен
            state = 2;//переходим к состоянию 2
         break;
      case 2: //Конец первого налива
         if(entry) //при входе в состояние
            {
               PauseGTimer(TIMER1); //Ставим таймер на паузу
               SetOut(KP, on);// Открываем клапан подачи
               SetOut(KN, off);// Закрываем клапан налива
               full_time = GetGTimer(TIMER1) * 10; //Вычисляем полное время налива
            };
         if(GetInput(DNU))//Если бак пуст
            state = 3;//переходим к состоянию 3
         break;
      case 3: //Заполнение бака
         if(entry) //при входе в состояние
            {
               ReleaseGTimer(TIMER1); //Снимаем таймер с паузы
               SetOut(KP, off);//Закрываем клапан подачи
               SetOut(KN, on);//Открываем клапан налива
            };
         if(GetInput(DVU))//Если бак заполнен
            state = 4;//переходим к состоянию 4
         if(GetGTimer(TIMER1) >= full_time)//Если время налива истекло
            state = 5;//переходим к состоянию 5
         break;
      case 4: //Опорожнение бака
         if(entry) //при входе в состояние
            {
               PauseGTimer(TIMER1); //Ставим таймер на паузу
               SetOut(KP, on);// Открываем клапан подачи
               SetOut(KN, off);// Закрываем клапан налива
            };
         if(GetInput(DNU))//Если бак пуст
            state = 3;//переходим к состоянию 3
         break;
      case 5: //Слив остатка воды
         if(entry) //при входе в состояние
            {
               SetOut(KP, on);// Открываем клапан подачи
               SetOut(KN, off);// Закрываем клапан налива
            };
         if(GetInput(DNU))Если бак пуст
            state = 6; переходим к состоянию 6
         break;
      case 6: //Конец налива и завершение работы
         if(entry) //при входе в состояние
            {
               SetOut(KP, off);//Закрываем клапан подачи
               SetOut(KN, off/ Закрываем клапан налива
            };
         state = 0;//Завершаем работу и переходим в неактивное состояние
         break;
   };
   state = state;
}
```

Сделаем некоторые пояснения к коду. В программе присутствует флаг entry, принимающий значение 1 на следующей итерации цикла после изменения автоматом состояния. Он нужен для того, чтобы действия в состояниях выполнялись однократно после входа автомата в состояние. В самом деле, если таймер будет сбрасываться на каждой итерации цикла, то он просто не будет работать! Для отслеживания изменения состояний служит вспомогательная переменная _state. После каждой итерации переменная _state приравнивается к переменной состояния автомата state.

О некоторых аспектах применения таймеров в автоматном программировании вы можете прочитать в [3–7].

На этом мы закончим обсуждение таймеров и перейдем к описанию некоторых практических примеров применения SWITCH-технологии.

Автор выражает глубокую благодарность Анатолию Абрамовичу Шалыто за ценные замечания и редактирование статьи.

Литература

1. Татарчевский В. А. Применение SWITCH-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Часть 3 // Компоненты и технологии. 2007. № 1.
2. Татарчевский В. А. Применение SWITCH-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Часть 4 // Компоненты и технологии. 2007. № 2.
3. Шахов В. Моделирование программно-аппаратных «реактивных» систем раскрашенными сетями Петри // RSDN Magazine 2006. № 3.
4. Петриковский А. Субъектное программирование // Компьютера. 2006. № 13.
5. Козаченко В. Ф. Эффективный метод программной реализации дискретных управляющих автоматов во встроенных системах управления. // www.motorcontrol.ru
6. Шалыто А. А., Туккель Н. И. SWITCH-технология — автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5.
7. Альтерман И. З., Шалыто А. А. Формальные методы программирования логических контроллеров // Промышленные АСУ и контроллеры. 2005. № 10.