Системный таймер

Зачастую возникает необходимость строгой привязки времени выполнения программы к реальному времени. Например, в случае, когда требуется сгенерировать сигнал нужной длительности. При этом настройка обычного таймера для решения подобной задачи слишком громоздка. В итоге для решения подобных задач в микроконтроллере предусмотрен специальный таймер реального времени. Включение этого таймера, вызывает возникновение прерываний через строго определённые промежутки времени, задаваемые самим разработчиком. В функцию, вызываемую этим прерыванием, вставляется код, который должен выполняться через заданные интервалы времени. Например, это может быть запуск аналогово-цифрового преобразования, т.к. зачастую необходимо, что бы частота дискретизации сигнала была стабильной. Остальная же часть функционирования таймера скрыта. Это позволяет не отвлекаться на настройку таймера и больше времени отвести для решения непосредственно поставленной задачи.

Кроме того, на системном таймере могут отсчитываться заданные промежутки времени любой длительности. Подробнее об этом будет рассказано во второй части методического пособия.

Инициализация системного таймера

Задержку программы можно задавать в тактах, но такую задержку сложно пересчитать в реальное время. Привязку к реальному времени можно сделать с помощью упомянутого ранее таймера реального времени.

Инициализация таймера производится следующей строкой:

```
SysTick_Config(SystemCoreClock /1000);//1ms
```

Число 1000 говорит сколько раз за секунду будет срабатывать прерывание от этого таймера. В нашем случае период работы таймера равен 1 мс.

В переменной SystemCoreClock записана реальная частота тактирования микроконтроллера. Она может отличаться от HSE, т.к. в микроконтроллере есть так называемая PLL, которая может умножать и делить тактовую частоту в заданное число раз. Чтобы узнать текущее реальное значение тактовой частоты микроконтроллера используется функция:

```
SystemCoreClockUpdate();
```

Рекомендуется использовать её до инициализации системного таймера.

При срабатывании прерывания от SysTick, будет вызываться функция:

void SysTick_Handler(void);

В эту функцию надо писать то, что должно произойти в момент возникновения прерывания.

Выглядит описание этой функции так:

```
void SysTick_Handler(void)
{
//Тут пишется код, который будет исполняться каждый вызов функции.
}
```

К примеру, рассмотрим алгоритм реализации функции задержки с помощью этого таймера:

Наша функция должна принять входной параметр, говорящий, на сколько миллисекунд должна быть произведена задержка. Функция первым делом копирует значение в глобальную переменную, которую системный таймер уменьшает каждый цикл прерывания. Теперь функции остаётся только подождать пока переменная не станет равной нулю. После чего завершить свою работу.

Алгоритм реализации длительных временных задержек

Зачастую возникает потребность в организации длительных задержек, кратных времени срабатывания системного таймера. Для организации таких задержек создаётся переменная, в которую будет записываться время задержки. При этом в прерывании системного таймера записывается код:

```
if (a>0)
{
        a--;
}
```

Когда же переменная a достигает значения, равного 1, то выставляется флаг. Если выставлять флаг при равенстве a нулю, то он будет выставляться каждое срабатывание прерывания системного таймера, пока в a не будет записано значение отличное от 0.

В итоге в функции прерывания имеем код вида:

```
if (a>0)
{
	a--;
	if (a == 1)
	{
		Flag=1;
	}
}
```

В самой программе проверяется флаг и по его установке выполняется какое-либо действие. Например инвертируется состояние светодиода

```
while(1)
{
	if (Flag==1)
	{
		Flag = 0;
		GPIO_ToggleBits(GPIOA, GPIO_Pin_8);
	}
}
```

Светодиод поменяет своё состояние после установки флага, как только программа дойдёт до этого места. Очевидно, что это может быть не сразу, как только установится флаг. Если же нужна мгновенная реакция, то инверсию необходимо прописать прямо в функции прерывания. Но стоит помнить, что при длительном выполнении функции прерывания есть шанс пропустить другое прерывание.

Либо можно аналогичный код писать прямо в функции прерывания. Например, если требуется мигать светодиодом, то можно в функции прерывания написать так:

```
if (i > 0)
{
	i--
} else {
	GPIO_ToggleBits(GPIOD, GPIO_Pin_12);
	i=1000;
}
```

Как только произойдёт 1000 прерываний, переменная i обнулится и состояние GPIOD.12 поменяется. Далее цикл начнётся заново.