AVR. Учебный курс. Конечный автомат

easyelectronics.ru/avr-uchebnyj-kurs-konechnyj-avtomat.html

DI HALT

Каждый кто пытался разбираться с конечными автоматами наверняка натыкался на всякие замудреные графы, какие то графики. Многие посчитав это слишком сложным плюнули и забили. А Зря!

С простейшим конечным автоматом каждый из нас сталкивался с самого детства — это механическая авторучка. Объект с единственной функцией «Нажатие кнопки», но в зависимости от очередности результат разный. Стержень то прячется, то вылазит.

Так и в нашем случае — конечный автомат это функция которая запоминает свое состояние и при следующем вызове делает свое черное дело исходя из прошлого опыта. Простой пример — мигалка (псевдокод):

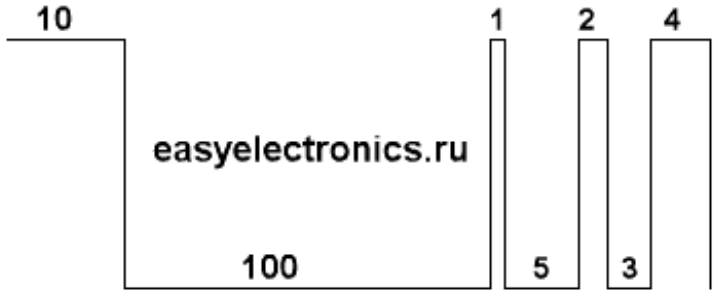
```
; Глобальные
переменные
u08 Blink_State;

void Blink(void)
{
if (Blink_State == 1)
 {
 Led_On();
 Blink_State = 0;
 Return;
 }
if (Blink_State == 0)
 {
 Led_Off();
 Blink_State = 1;
 Return;
 }
}
```

Вызывая эту функцию подряд мы заставим диодик менять свое состояние при каждом вызове.

Но никто не мешает взять и сделать автомат куда сложней, на несколько десятков состояний. Никто не запрещает использовать вложенные конечные автоматы, никто не запрещает менять состояние автомата извне. В общем, мощнейший инструмент для построения быстрых и очень компактных алгоритмов.

Приведу другой, более сложный пример. Генерацию сигнала сложной формы.

Пусть у нас есть сигнал:



Цифрами указана задержка, в каких нибудь величинах. Это не суть важно.

Обычно народ не парится и лепит его по быдлокодерски, через delay:

```
Set_Pin(); // Вывод в 1
Delay(10); // Задержка
10

Clr_Pin(); // Вывод в 0
Delay(100); // Задержка
в 100

Set_Pin();
Delay(1);

Clr_Pin();
Delay(5);

Set_Pin();
Delay(2);

Clr_Pin();
Delay(3);

Set_Pin;
Delay(4);

Clr_Pin();
```

Это дает минимальный код, но затыкает работу контроллера на весь период посылки. А если слать надо постоянно? Да еще дофига всего попутно делать? Экран обновлять, данные обрабатывать, в память писать...

В таком случае у нас рулит RTOS, где на Delay происходит передача управления диспетчеру. Но если ОС нету? Вот тут то и идет в ход конечный автомат.

Проще всего тут применить таймер и его прерывание по переполнению. Таймер сам, в своем прерывании, будет загружать себя новыми значениями выдержки и щелкать дальше.

```
Timer_Overflow_Interrupt(void)
{
switch(TMR_State)   // Обработчик прерывания по
переполнению
 {
 case 0:
  {
  Clr_Pin();  // Вывод в 0
  TCNT = 255-100;  // Задержка в 100 (до переполнения)
  TMR_State = 1;   // Следующая стадия 1
  Break;   // Выход
  }

 case 1:
  {
  Set_Pin();
  TCNT = 255-1;
  TMR_State = 2;
  Break;
  }

 case 2:
  {
  Clr_Pin();
  TCNT = 255-5;
  TMR_State = 3;
  Break;
  }

 case 3:
  {
  Set_Pin();
  TCNT = 255-2;
  TMR_State = 4;
  Break;
  }

 case 4:
  {
  Clr_Pin();
  TCNT = 255-3;
  TMR_State = 5;
  Break;
  }

 case 5:
  {
  Set_Pin();
  TCNT = 255-4;
  TMR_State = 6;
  Break;
  }

 case 6:
  {
  Clr_Pin();
  Timer_OFF();   // Выключаем таймер. Работа окончена
  TMR_State = 0;   // Обнуляем состояние
  Break;
  }

 default:  break;
 }
}
```

А запускается эта байда простым пинком:

```
Set_Pin();
TCNT=255-10;  // Грузим таймер на выдержку 10
тиков
TMR_State = 0;  // Устанавливаем начальное
положение
Timer_ON();  // Поехали!

... // После чего можно заниматься чем угодно.
```

Когда автомат отработает — сама себя выключит. Можно еще какоенить событие сгенерировать, дабы основная программа поняла, что все уже готово. Хотя основная программа спокойно может палить переменную состояния автомата и все оттуда узнать сама.

Конечный автомат можно зациклить — скажем на стадии 6 сделать перенаправление на стадию 0, то получим генератор сигнала сложной формы. Причем он будет занимать минимум процессорного времени.

Если мы, например, захотим сделать десяток ШИМ сигналов? То что нам мешает повесить их на ОДИН таймер, главное отсортировать все скважности по возростанию, причем сортировать их вместе с ногами которыми нужно дрыгать. А потом прогнать по прерыванию таймера конечный автомат, да так чтобы он по стадиям передергивал ножки. Правда при изменении скважности любого из этих софтверных ШИМ каналов придется делать повторную сортировку всего массива. Разумеется больших скоростей мы на этом не получим. Таймер не может щелкать так быстро, да и математики там хватит. Но для многих задач, например, одновременное управление десятком сервомашинок этого более чем достаточно.

А если в том же прерывании таймера сделать выбор следующей стадии исходя не из тупой последовательности, а, скажем, на основе обрабатываемого байта, то мы получим программно-аппаратный генератор, к примеру, 1-Wire кода. Достаточно анализировать входной буфер и если там 1 — перебрасывать автомат в состояние соответствующее отработки генерации сигнала 1, а если 0, то в состояние генерящее выдержку нуля.

Более того, на автоматах можно полностью построить логику работы программы. От и до. Получается весьма торчково, пока не въедешь в это всей душой мозг сломать можно, но зато работает просто зверски и очень проста в отладке.

Не буду описывать это так как есть замечательный цикл статей Владимира Татарчевского, опубликованный в журнале Компоненты и Технологии. Настоятельно рекомендую к прочтению. Чтобы вам не шерстить по инету, я все части этой замечательной статьи выложил в **архив**. Качайте и вкуривайте.

Особенно автоматный метод доставляет тем, что готовые автоматы вписываются как родные в ЛЮБУЮ почти архитектуру. У меня они и во флаговых автоматах работают на ура, и из диспетчера RTOS я их гоняю как родные. Красота!