**Лабораторная работа № 3**

**Цель работы:** Изучение основных особенностей работы с таймерами через прерывания при программировании для микроконтроллеров (МК) ARM.

**Приборы и материалы:**

1. Отладочная плата MDR1986VE91T Rev 4
2. Программатор J-Link ARM
3. Блок питания 5В, 1.4А
4. ПК с установленной средой программирования Keil uVision

**Ход работы**

1. Собрать аппаратную часть по рекомендациям лабораторной работы № 2
2. Открыть проект MDRProject в среде программирования Keil uVision.
3. Подключить к проекту библиотеку TIMER, необходимую для работы с таймерами.

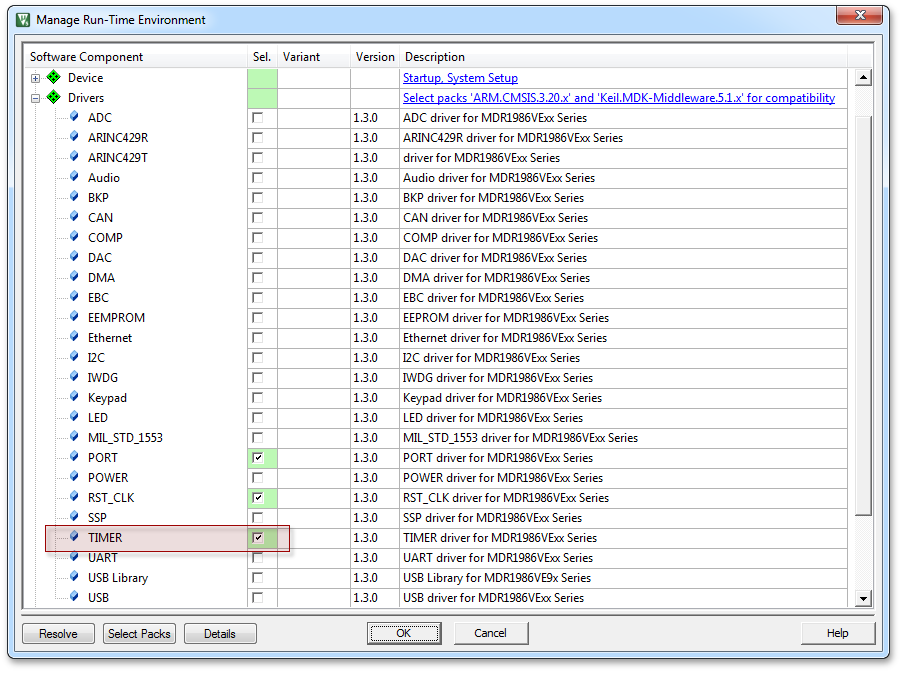


Рисунок 1 – Библиотеки проекта для работы с таймерами

1. Добавить в заголовок основного файла исходного кода *main.c* ссылку на заголовочный файл библиотеки TIMER:

#include <MDR32F9Qx\_timer.h>

1. Вынести содержимое главного цикла в отдельную функцию, которая при каждом вызове переключает светодиод на следующий:

uint8\_t cur\_i;

void NextLED(){

switch(cur\_i++ % 5) {

case 0:

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_14);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_10);

break;

case 1:

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_10);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_11);

break;

case 2:

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_11);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_12);

break;

case 3:

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_12);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_13);

break;

case 4:

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_13);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_14);

break;

}

}

1. Инициализировать переменную cur**\_i** нулём в функции **main()** до главного цикла, чтобы при первом вызове директиве **cur\_i++**, было к чему прибавлять единицу.
2. Создать функцию TimerInit(), в которой производится инициализация аппаратного таймера TIMER1 и соответствующего прерывания по переполнению, и вызвать ее в функции **main()** до главного цикла:

//Объявление инициализационной структуры

TIMER\_CntInitTypeDef TIM1Init;

void TimerInit(){

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_TIMER1, ENABLE);

//Установка первого делителя тактовой частоты таймера

TIMER\_BRGInit(MDR\_TIMER1, TIMER\_HCLKdiv1);

//Загрузка значений по-умолчанию в структуру TIM1Init

TIMER\_CntStructInit(&TIM1Init);

TIM1Init.TIMER\_Prescaler = 8000; // Второй делитель частоты

TIM1Init.TIMER\_Period = 1000; // Период переполнения

TIMER\_CntInit(MDR\_TIMER1, &TIM1Init);

//Настройка прерывания

NVIC\_EnableIRQ(Timer1\_IRQn); //Включение прерываний от TIMER1

NVIC\_SetPriority(Timer1\_IRQn, 0); //Установка приоритета

//Включение прерывания при равенстве нулю значения TIMER1

TIMER\_ITConfig(MDR\_TIMER1, TIMER\_STATUS\_CNT\_ZERO, ENABLE);

//Запуск таймера

TIMER\_Cmd(MDR\_TIMER1, ENABLE);

}

1. Написать обработчик прерывания TIMER1, в котором вызвать функцию NextLED().

void Timer1\_IRQHandler() {

//Проверка что причина прерывания – обновление таймера

if(TIMER\_GetITStatus(MDR\_TIMER1, TIMER\_STATUS\_CNT\_ZERO)){

NextLED();

//Очистка флага прерывания (это необходимо делать в конце)

TIMER\_ClearITPendingBit(MDR\_TIMER1, TIMER\_STATUS\_CNT\_ZERO);

}

}

1. Загрузить и отладить программу по рекомендациям в лабораторной работе № 2.

**Теоретические сведения**

Рисунок 2 иллюстрирует работу таймера в микроконтроллерах. После запуска, таймер начинает прибавлять к начальному значению счетчика фиксированное число через фиксированный промежуток времени, который устанавливается двумя делителями тактовой частоты МК. При достижении таймером значения периода, таймер «Обновляется» и сбрасывается в 0. Можно настроить прерывание, возникающее при данном событии. Таким образом, в коде выше, общий делитель частоты равен 8000, что при делении на тактовую частоту МК а 8 МГц, дает смену значения счетчика 1000 раз в секунду. Период установлен в 1000, что позволяет генерировать прерывания каждую секунду.

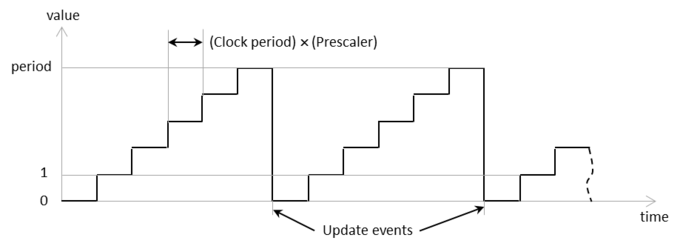


Рисунок 2 – Иллюстрация работы таймера

Преимущество такого способа отсчета временных интервалов в том, что главный цикл при этом пуст и МК в промежутке между прерываниями может выполнять любой другой код. В идеале, программа для МК должна быть организована таким образом, чтобы прерывания меняли переменную состояния, а главный цикл ее проверял и выполнял различные действия в зависимости от текущего состояния.