**8 ЛАБОРАТОРНАЯ РАБОТА №7**

**СИСТЕМА ПРЕРЫВАНИЙ МИКРОКОТРОЛЛЕРА ATMEGA8515. ПРЕРЫВАНИЯ ОТ ТАЙМЕРА**

**Цель работы**

Ознакомиться с функциональными возможностями и внутренней структурой отладочного стенда EV8031/AVR. Изучить внутреннюю организацию таймеров/счётчиков на примере микроконтроллера ATMega8515. Научиться конфигурировать и обрабатывать прерывания от таймера.

**8.1 Краткие теоретические сведения**

Для управления прерываниями от таймера/счётчика Т1 используются отдельные биты регистра TIMSK, формат которого показан на рисунке 7.10.

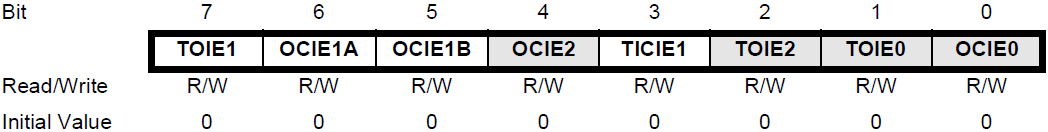


Рисунок 8.1 – Формат регистра TIMSK в МК ATMega8515 для Т1

Бит TOIE1 (если установлен в "1") разрешает прерывание по переполнению Т1, OCIE1A – по совпадению A.

Для индикации наступления прерываний от таймера/счётчика Т1 служит регистр флагов TIFR, формат которого показан на рисунке 7.11.

Флаг TOV1 индицирует переполнение таймера/счётчика Т1, OCF1A – совпадение A.

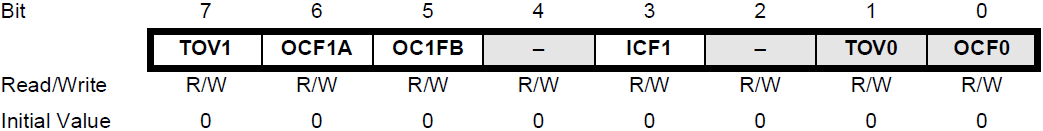


Рисунок 8.2 – Формат регистра TIFR в МК ATMega8515 для Т1

Как видно из рисунка 7.9, в состав таймера/счётчика Т1 входит 4 следующих основных регистра ввода/вывода (8- и 16-разрядных):

– TCCR1A, TCCR1B – регистры управления (8-разрядные);

– TCNT1 – счётный регистр (16-разрядный);

– OCR1A – регистр сравнения (16-разрядный).

Следует помнить, что каждый из 16-разрядных регистров физически разделен на 2 8-разрядных регистра. Названия этих 8-разрядных регистров образуются путём прибавления буквы "H" (старший байт) и "L" (младший) к названию 16-разрядного регистра. Например, 16-разрядный счётный регистр TCNT1 таймера/счётчика T1 физически расположен в паре 8-разрядных регистров TCNT1H:TCNT1L.

Счётный регистр TCNT1 входит в состав модуля реверсивного счётчика. В зависимости от режима работы, содержимое счётного регистра может обнуляться, инкрементироваться или декрементироваться по каждому импульсу тактового сигнала clkT1 (который поступает с предделителя). Регистр TCNT1 в любой момент времени доступен как для чтения, так и для записи. После подачи питания либо сброса этот регистр содержит нулевое значение. При достижении таймером/счётчиком максимального значения (0xFFFF) устанавливается флаг TOV1 в регистре TIFR. Для того, чтобы при этом сгенерировалось прерывание, должен быть установлен в "1" бит TOIE1 в регистре TIMSK, а также флаг I в регистре SREG.

Регистр сравнения OCR1A входит в состав блока сравнения таймера/счётчика Т1. Во время работы Т1 непрерывно (в каждом такте) производится сравнение содержимого регистров OCR1A и TCNT1. В случае равенства устанавливается флаг OCF1A в регистре TIFR. Для того, чтобы при этом сгенерировалось прерывание, должен быть установлен в "1" бит OCIE1A в регистре TIMSK, а также флаг I в регистре SREG.

Следует помнить, что любая операция записи в счётный регистр TCNT1 заблокирует работу блока сравнения и формирование сигнала о совпадении на 1 период тактового сигнала. В связи с этим рекомендуется перед изменением содержимого регистра OCR1A производить останов таймера/счётчика и сброс его счётного регистра TCNT1, затем таймер можно запустить снова.

Для управления работой таймера/счётчика Т1 предназначены регистры управления TCCR1A и TCCR1B, форматы которых показаны на рисунках 7.12 и 7.13 соответственно.

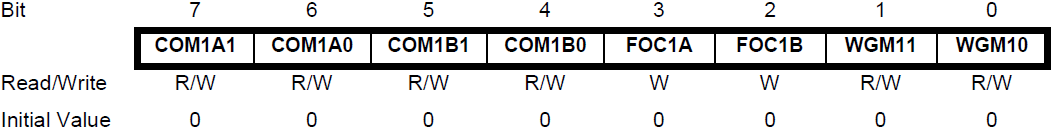


Рисунок 8.3 – Формат регистра TCCR1А в МК ATMega8515

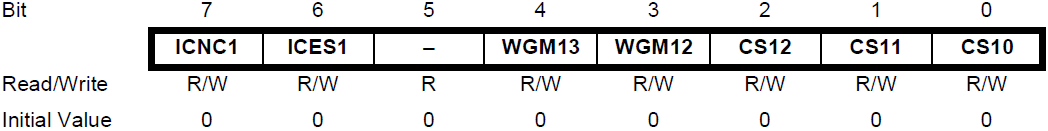


Рисунок 8.4 – Формат регистра TCCR1B в МК ATMega8515

Рассмотрим назначение только основных битов этих регистров.

Биты WGM13 – WGM10 отвечают за выбор режима работы таймера/счётчика Т1 в соответствии с таблицей, показанной на рисунке 7.14.

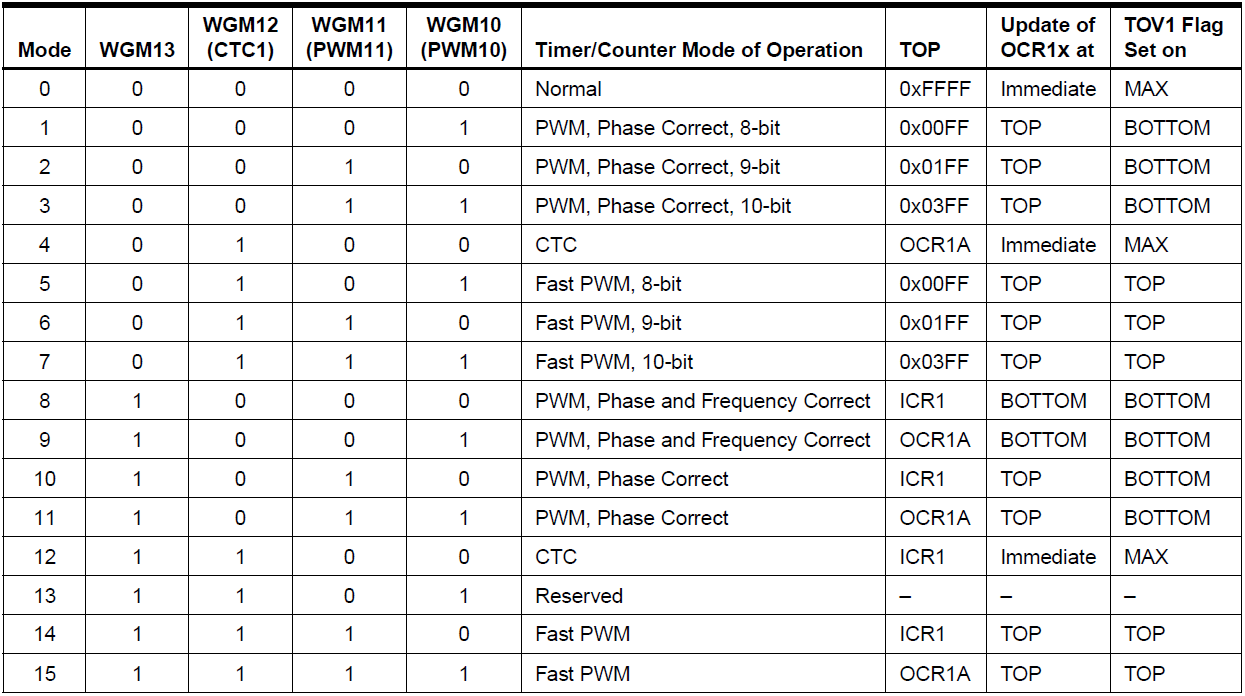


Рисунок 8.5 – Конфигурация режимов работы таймера/счётчика Т1

Биты CS12 – CS10 отвечают за выбор источника тактового сигнала для Т1 в соответствии с таблицей, показанной на рисунке 7.15.

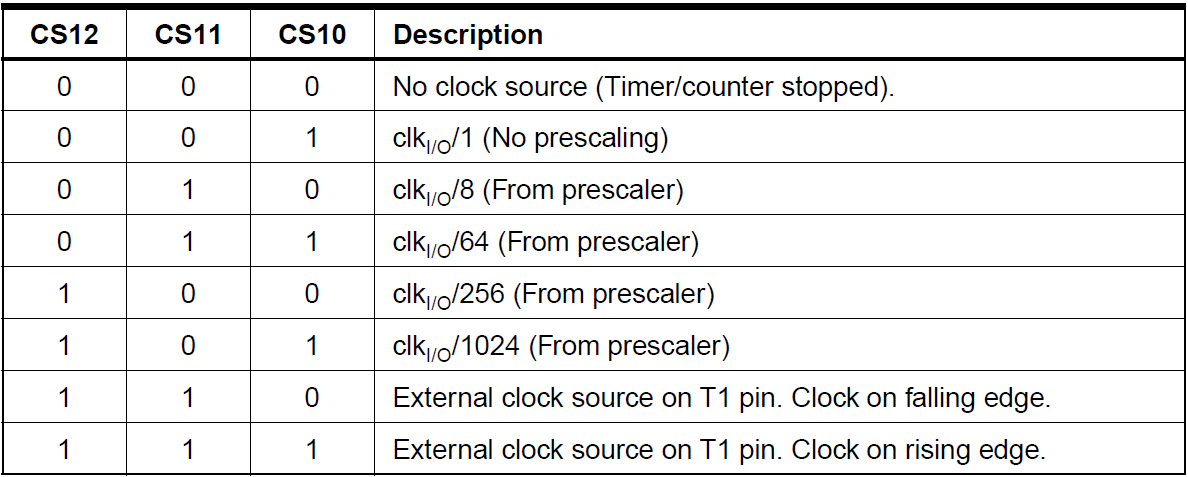


Рисунок 8.6 – Выбор тактового сигнала таймера/счётчика Т1

Следует помнить, что запись в биты CS12 – CS10 не только выбирает возможный источник тактового сигнала, но и производит пуск/останов таймера/счётчика Т1.

К наиболее часто используемым режимам работы таймера/счётчика Т1 следует отнести режим Normal и CTC.

**8.2 Порядок выполнения работы**

8.2.1 Запустить IDE AVR Studio 4.

8.2.2 Создать новый проект в IDE AVR Studio 4.

8.2.3 В появившемся окне написать программу на языке С или ассемблер с учётом варианта задания, который указан в таблице 7.1. Во всех вариантах подразумевается работа с прерываниями, которые генерируются при помощи таймера/счётчика T1.

Таблица 8.1 – Варианты заданий

|  |  |
| --- | --- |
| № | Описание задания |
| 10 | По нажатию кнопки SW16 осуществлять пуск секундомера, а по нажатию SW15 – его останов. Состояние секундомера отображать во второй строке ЖКИ. Организацию временных задержек выполнять при помощи таймера/счётчика Т1. Обработку нажатий кнопок осуществлять при помощи механизма внешних прерываний. |

8.2.4 Произвести компиляцию проекта.

8.2.5 При наличие сообщений об ошибках или предупреждениях вернуться к предыдущему пункту и внести необходимые исправления. В случае некорректной работы программы выполнить её отладку средствами меню Debug.

8.2.6 Проверить подключение USB-кабеля программатора к одноимённому разъёму системного блока.

8.2.7 Загрузить исполняемый файл проекта в микроконтроллер.

8.2.8 Визуально оценить правильность работы написанной программы.

**8.3 Исходный текст программы**

#include <avr/io.h>

#include <avr/iom8515.h>

#include <avr/interrupt.h>

#include <util/delay.h>

//çàäà¸ì êîíñòàíòó - àäðåñ ðåãèñòðà êîìàíä â ÆÊÈ

#define LCD\_COMMAND 0x8004

//çàäà¸ì êîíñòàíòó - àäðåñ ðåãèñòðà äàííûõ â ÆÊÈ

#define LCD\_DATA 0x8005

#define TRUE 1

#define FALSE 0

volatile unsigned char end\_time\_interval= FALSE;

volatile char stop = FALSE;

volatile unsigned char seconds = 0;

ISR(TIMER1\_COMPA\_vect){

end\_time\_interval = TRUE;

}

//knopka sw15 reset

ISR(INT0\_vect){

seconds = 0;

}

//knopka sw16 pause/start

ISR(INT1\_vect){

stop = stop ? FALSE : TRUE;

}

void hardware\_delay\_ms(int koef){

TIMSK = 1 << OCIE1A;

OCR1A = koef; //287

SFIOR = 1 << PSR10;//27647

TCCR1A = 0x00;

TCCR1B = 1 << WGM12 | 1 <<CS12;

while(!end\_time\_interval){

};

end\_time\_interval = FALSE;

TIMSK = 0;

}

int main(void){

//íà÷àëüíàÿ èíèöèàëèçàöèÿ êîíòðîëëåðà

//ðàçðåøàåì ðàáîòó ñ âíåøíåé ïàìÿòüþ (îáðàùåíèå ê ñèñòåìíîìó êîíòðîëëåðó ñòåíäà)

MCUCR = 1<<SRE | 1 << ISC00 | 1 << ISC01 | 1 << ISC10 | 1 << ISC11;

//îòêëþ÷åíèå ïèòàíèÿ àíàëîãîâîãî êîìïàðàòîðà

ACSR = 1<<ACD;

GICR = 1<<INT1 | 1<<INT0;

//îïðåäåëÿåì ìàñêè äëÿ çàæèãàíèÿ ñâåòîäèîäîâ

//óñòàíàâëèâàåì óêàçàòåëü íà àäðåñ ðåãèñòðà êîìàíä ÆÊÈ

volatile unsigned char \*command;

command= (unsigned char\*) LCD\_COMMAND;

//óñòàíàâëèâàåì óêàçàòåëü íà àäðåñ ðåãèñòðà äàííûõ ÆÊÈ

volatile unsigned char \*data;

data= (unsigned char\*) LCD\_DATA;

// \_delay\_ms(10);

sei();

hardware\_delay\_ms(27647 ); //óñòàíàâëèâàåì çàäåðæêó ïî âêëþ÷åíèþ ïèòàíèÿ íà 1 ñ

//íà÷àëüíàÿ èíèöèàëèçàöèÿ ÆÊÈ

\*command= 0x38; //Function Set

// \_delay\_ms(10);

hardware\_delay\_ms(287); //âûçûâàåì çàäåðæêó íà 10 ìñ

\*command= 0x0E; //Display On/Off Control

// \_delay\_ms(10);

hardware\_delay\_ms(287); //âûçûâàåì çàäåðæêó íà 10 ìñ

\*command= 0x01; //Display clear

//\_delay\_ms(10);

hardware\_delay\_ms(287); //âûçûâàåì çàäåðæêó íà 10 ìñ

\*command= 0x06; //Entry Mode Set

\_delay\_ms(10);

hardware\_delay\_ms(287); //âûçûâàåì çàäåðæêó íà 10 ìñ

//âûâîä ïåðâîé ñòðîêè ñîîáùåíèÿ

\*command = 0xC0;

\_delay\_ms(10);

// hardware\_delay\_ms(287);

volatile unsigned char x2 = 0;

volatile unsigned char x1 = 0;

volatile unsigned char x0 = 0;

while(1){ //áåñêîíå÷íûé öèêë

if (!stop){

x2 = seconds / 100;

x1 = (seconds - x2\*100) / 10;

x0 = seconds - x2\*100 - x1\*10;

\*data = 0x30 | x2;

// \_delay\_ms(100);

hardware\_delay\_ms(2870);

\*data = 0x30 | x1;

\_delay\_ms(10);

hardware\_delay\_ms(2870);

\*data = 0x30 | x0;

// \_delay\_ms(10);

hardware\_delay\_ms(2870);

\*command = 0xC0;

// \_delay\_ms(10);

hardware\_delay\_ms(27647);

++seconds;

}

}

return 0;

}

**8.4 Особенности IDE AVRStudio выявленные в ходе выполнения лабораторной работы**

В ходе лабораторной работы никаких новых особенностей IDE AVRStudio не было выявлено.

**Выводы**

В ходе лабораторной работы были изучены принципы работы с таймером в микроконтроллерах семейства AVR. Была написана программа, которая реализует секундомер с индикацией на жидкокристаллическом индикаторе, по нажатию на кнопку SW16 происходит приостанов и запуск секундомера, по нажатию на SW15 происходит сброс секундомера, все задержки реализованы аппаратно с помощью таймера.