**Лабораторная работа 5**

**Использование таймера TIMER0 в режиме прерываний**

Мощевикин А.П., ПетрГУ

***Общие сведения о таймерах-счетчиках в микроконтроллере Комдив-МК***

Микроконтроллер Комдив-МК содержит восемь 64-разрядных таймеров/счетчиков общего назначения, один специализированный сторожевой таймер, а также таймер системного времени – таймер временной метки TS (Time Stamp) [5].

Базовые адреса всего блока таймеров и каждого таймера в отдельности определены в файле ./psp\_mc/arch/kmk/uke.h.

#define TIMER\_BL\_BASE 0xbb503000

#define TIMER0\_BASE 0xbb503020

#define TIMER1\_BASE 0xbb503040

#define TIMER2\_BASE 0xbb503060

#define TIMER3\_BASE 0xbb503080

#define TIMER4\_BASE 0xbb5030a0

#define TIMER5\_BASE 0xbb5030c0

#define TIMER6\_BASE 0xbb5030e0

#define TIMER7\_BASE 0xbb503100

Для выполнения этой лабораторной работы имеет смысл определить еще и адрес регистра Cause – регистра источника выходных прерываний [1] (примечание: с точки зрения контроллера прерываний регистр Cause правильнее было бы назвать регистром источника запросов на прерывание, а не выходных прерываний).

#define TIMER\_CAUSE 0xbb503008

В младших восьми разрядах (поле INT) регистра Cause содержатся битовые флаги выставленных запросов на прерывание от таймеров TIMER0..TIMER7.

Взаимодействие с таймерами осуществляется через 5 регистров (в скобках указано смещение относительно базового адреса каждого таймера) [1]:

Control (32-битный регистр, смещение 0х00) – регистр управления,

Status (32-битный регистр, смещение 0х04) – регистр состояния,

Load (64-битный регистр, смещение 0х08) – регистр начального значения счетчика,

Read (64-битный регистр, смещение 0х10) – регистр текущего значения счетчика,

PWM (64-битный регистр, смещение 0х18) – регистр настройки ШИМ.

Макросы ниже задают смещения этих регистров.

#define TIMER\_CTRL\_REG\_OFFS 0x00

#define TIMER\_STATUS\_REG\_OFFS 0x04

#define TIMER\_LOAD\_REG\_OFFS 0x08

#define TIMER\_READ\_REG\_OFFS 0x10

#define TIMER\_PWM\_REG\_OFFS 0x18

Битовую структуру регистра управления Control и регистра состояния Status можно представить следующими определениями.

#define TIMER\_CTRL\_CNT\_EN 0 // 1 – счет разрешен, 0 – счет запрещен

#define TIMER\_CTRL\_MODE 1 // 1 – циклический режим, 0 – одиночный

#define TIMER\_CTRL\_INT\_EN 2 // 1 – прерывание включено

#define TIMER\_CTRL\_RST\_EN 3 // 1 – выполнить RESET микроконт-ра после окончания счета

#define TIMER\_CTRL\_OUT\_POL 4 // полярность сигнала при использовании GPIO в ШИМ-режиме

#define TIMER\_CTRL\_PWM\_EN 5 // включение линии GPIO в ШИМ-режиме

#define TIMER\_CTRL\_MEA\_EN 6 // разрешение меандра

#define TIMER\_CTRL\_CLK\_SEL 7 // 1 – внутренний сигнал 100 МГц, 0 – внешний сигнал

#define TIMER\_CTRL\_SYNC\_EN 8 // 1 – синхронизация включена

#define TIMER\_CTRL\_EXT\_CLK 9 // 1 – внешнее разрешение счета

#define TIMER\_CTRL\_DIV 16

#define TIMER\_STATUS\_ST 0

В поле DIV (разряды 26..16) заносится значение предделителя тактовой частоты. Если значение этого поля равно 0x7FF, то таймер не будет использовать предделитель: значение счетчика будет уменьшаться на единицу каждые 10 наносекунд (при тактировании от внутреннего сигнала с частотой 100 МГц).

В последней строке обозначен номер бита ST (состояние) в регистре Status. Запись «1» в этот разряд снимает запрос на прерывание от счетчика.

Все таймеры считают вниз (к нулю), начиная с предустановленного значения. Когда значение счётчика достигает нуля, значение бита ST регистра Status устанавливается в «1», возникает запрос на прерывание. Если для таймера разрешен запрос на прерывание, то соответствующий номеру таймера бит в поле INT регистра Cause также устанавливается в значение «1». Биты в полях регистра Cause сбрасываются автоматически при снятии прерываний в регистре Status таймеров (необходимо записать «1» в поле ST).

Для того, чтобы запустить таймер-счетчик TIMER0 в циклическом режиме с прерываниями каждую миллисекунду (счет на частоте 100 МГц без предделителя), необходимо выполнить следующую последовательность действий [1].

1. Остановить таймер – снять разрешение на счет, записав «0» в поле CNT\_EN регистра Control соответствующего таймера.
2. Проверить состояние бита ST в регистре Status и сбросить предыдущий запрос на прерывание, если он был установлен, записав «1» в поле ST.
3. Установить начальное значение счетчика 100000, записав его в регистр Load. Значение 100000 соответствует периоду 1 мс для тактовой частоты 100 МГц.
4. Запустить счетчик, выставив значения в регистре Control: DIV=0x7FF, CLK\_SEL=1, INT\_EN=1, MODE=1, CNT\_EN=1.

На языке Си этот фрагмент исходного кода приведен ниже.

char s[100];

unsigned long temp;

\*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_CTRL\_REG\_OFFS) = \

 (0x7FF << TIMER\_CTRL\_DIV) | \

 (1 << TIMER\_CTRL\_CLK\_SEL) | \

 (1 << TIMER\_CTRL\_INT\_EN) | \

 (1 << TIMER\_CTRL\_MODE);

temp = \*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_STATUS\_REG\_OFFS);

//\_\_print\_console("TIMER\_STATUS = ");

//\_\_print\_console(itoa(temp, s, 10));

//\_\_print\_console("\r\n");

\*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_STATUS\_REG\_OFFS) = \

 (1 << TIMER\_STATUS\_ST);

\*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_LOAD\_REG\_OFFS) = 100000; // 1 ms

\*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_CTRL\_REG\_OFFS) = \

 (0x7FF << TIMER\_CTRL\_DIV) | \

 (1 << TIMER\_CTRL\_CLK\_SEL) | \

 (1 << TIMER\_CTRL\_INT\_EN) | \

 (1 << TIMER\_CTRL\_MODE) | \

 (1 << TIMER\_CTRL\_CNT\_EN);

Операция во второй строчке описывает первое действие: остановить таймер TIMER0, если он работал.

Конструкция (0x7FF << WDT\_CTRL\_DIV) | (1 << WDT\_CTRL\_CLK\_SEL) | (1 << TIMER\_CTRL\_INT\_EN) | (1 << TIMER\_CTRL\_MODE\_EN) формирует значение, которое записывается в регистр Control: выставляются биты CLK\_SEL в седьмом разряде (нумерация с нуля), INT\_EN во втором разряде, MODE в первом разряде, и значение 0x7FF, которое сдвигается влево на 16 разрядов. Значение поля CNT\_EN остается равным 0.

Конструкцию \*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_CTRL\_REG\_OFFS) следует трактовать так. Сначала в правых скобках рассчитывается адрес регистра Control: к базовому адресу 0xbb503020 прибавляется смещение 0x00. Затем в левых скобках указывается, что, начиная с этого адреса, хранится беззнаковое четырехбайтовое целое число. Модификатор volatile используется, чтобы указать компилятору не оптимизировать использование памяти для данной операции. Например, не пытаться организовать регистровую переменную. Процессорное ядро обязательно должно провести запись в память по этому адресу: передать данные в контроллер блока таймеров. Самая левая звездочка в коде указывает на саму операцию: взятие содержимого по адресу. Таким образом, выставленные биты в составе 32-битного числа будут записаны в регистр Control таймера TIMER0, поскольку этот регистр «закреплен» за этим адресом в линейном пространстве памяти.

Необходимо обратить внимание (см. предпоследнюю строчку), что в 64-разрядный регистр Load записывается 32-разрядное значение типа unsigned long, а не 64-разрядная величина unsigned long long. Поскольку число 100000 не превышает максимальное значение 4-байтного числа, его запись можно провести только в нижнюю половину 64-разрядного регистра Load. Если бы записываемое число превышало 232, старшие биты числа должны были быть записаны в верхнюю половину регистра Load.

Выполнение кода выше (например, внутри функции init\_board()) приведет к тому, что каждую миллисекунду таймер-счетчик TIMER0 будет циклически начинать свой счет со значения в регистре Load и доходить до нуля, взводя бит (флаг) запроса на прерывание ST в регистре Status.

Чтобы периодически выполнять какое-то действие, привязанное к окончанию счета таймера, требуется настроить контроллер прерываний и создать обработчик прерывания (см. лабораторную работу №4). Необходимо отметить, что в обработчике должен быть обязательно сброшен флаг запроса на прерывание от сработавшего таймера, иначе последующие запросы от него не будут обрабатываться.

\*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_STATUS\_REG\_OFFS) = \

 (1 << TIMER\_STATUS\_ST);

***Действия программиста по настройке и использованию прерываний от таймера TIMER0***

Для использования прерываний от таймера TIMER0 в своей программе, программист должен выполнить следующее.

1. Создать функцию-обработчик запроса на прерывание от таймера TIMER0, например,

void my\_timer0\_int\_handler(unsigned char param) {

 . . .

}

… и связать ее с 8-ой линией входных запросов IRQ (8-ая линия – это IRQ\_TIMER0, см. файл ./psp\_mc/arch/kmk/uke.h).

int\_connect(IRQ\_TIMER0, my\_timer0\_int\_handler, 0);

1. Настроить TIMER0 на использование прерываний (см. раздел выше).
2. Объявить функцию обработки прерывания INT, привязать ее к одному из шести прерываний процессора INT0..INT5 (вместо INT\_NUMBER подставить, например, 2).

void kmk\_my\_int2(void);

kmk\_set\_int\_vector(INT\_NUMBER, &kmk\_my\_int2);

1. Настроить внутреннюю маршрутизацию контроллера прерываний (заполнить поля маски и приоритета, а также однозначно связать номер входного запроса на прерывание IRQ\_NUMBER с выходной линией прерывания процессора INT\_NUMBER).

kmk\_init\_int\_line(IRQ\_TIMER0, INT\_NUMBER, 0);

***Задания для самостоятельной работы***

Разработчики платы БАГЕТ-ПЛК1-01 [2] рекомендуют использовать среду разработки и структуру проекта, описанные в [3] и [4, раздел 3].

Задания к лабораторной работе подразумевают, что работа выполняется в ОС Windows с предустановленной виртуальной машиной WSL2 с Debian [3] и ППП МК [4-5]. Проекты на языке Cи следует создавать в папке ./psp\_mc/apps/ .

1. Используя исходный код лабораторной работы 1 для проекта на языке Си, создать программу, которая заставляет светодиод VD2 на плате БАГЕТ-ПЛК1-01 бесконечно мигать с периодичностью в 0.5 секунды.

Структура программы с пустым циклом внутри представлена ниже.

int main(void) {

 init\_board();

 while (1) {

 }

 return 1; // unreacheable

}

В функции init\_board() инициализировать таймер TIMER0 на работу с прерываниями. Создать функцию-обработчик прерываний, внутри которой изменяется состояние светодиода (горит / не горит). Каждый раз при вызове обработчика прерывания по таймеру оно должно изменяться на противоположное (для этого можно использовать глобальную переменную, хранящую состояние светодиода). Не забыть в обработчике сбросить запрос на прерывание от его источника.

1. Факультативно. Переработать программу из п.1. Добавить обработчик прерывания от кнопки SW2. Сделать так, чтобы при очередном нажатии кнопки SW2 запрещалось или разрешалось мигание светодиода VD2.

Примечание. При работе одновременно с двумя прерываниями необходимо выставить у них разные приоритеты (третий параметр функции kmk\_init\_int\_line(int IRQ, int INT, int priority)).

Для вывода результата в консоль можно воспользоваться функцией \_\_print\_console(). Для вывода целочисленных значений необходимо создать функцию перевода числа в строку, либо воспользоваться сторонней библиотекой, содержащей функцию типа itoa().

***Список литературы***

[1] Микросхема интегральная К5500ВК018. Указания по применению. Чаcть 5. Блок таймеров. ЮКСУ.431295.019Д4.4 // К5500ВК018\_d4\_04\_BL\_TIMER\_v.1.5.doc

[2] ПЛК «БАГЕТ-ПЛК1-01» Руководство по эксплуатации. ЮКСУ.421457.002-01РЭ. https://www.niisi.ru/БАГЕТ-ПЛК1-01\_РЭ\_v3.3.pdf

[3] Мощевикин А.П., Голяков М.А. Установка среды разработки в Windows 10 для БАГЕТ-ПЛК1-01 // BAGET-PLK1-01\_getting\_started\_v\_2\_X.doc

[4] Пакет поддержки программирования микроконтроллера (ППП МК). Описание применения. ЮКСУ.91264-01 31 01 // ППП\_МК\_31\_Описание\_применения v12.docx

[5] Пакет поддержки программирования микроконтроллера (ППП МК). Руководство программиста. ЮКСУ.91264-01 33 01 // ППП\_МК\_33\_Руководство\_программиста\_v5.docx