**Лабораторная работа 2**

**Таймер временной метки TS**

Мощевикин А.П., ПетрГУ

Разработчики платы БАГЕТ-ПЛК1-01 [1] рекомендуют использовать среду разработки и структуру проекта, описанные в [2] и [3, раздел 3]. Упомянутые ниже файлы входят в пакет поддержки программирования микроконтроллера ППП МК [3, 4].

***Общие сведения о таймере временной метки 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].

Существует два способа организации учета системного времени.

Первый реализуется с помощью таймера/счетчика общего назначения. В его настройках задается предел счета, равный, например, 10 микросекундам. Также организуется глобальная переменная, которая будет хранить системное время в микросекундах. По окончании счета таймер генерирует аппаратное прерывание. В обработчике прерываний значение переменной системного времени увеличивается на 10 и снова запускается таймер. Если пользователю понадобилось узнать текущее время, то он просто считывает значение переменной. Прерывания от таймера должны быть отключены на момент чтения, иначе возможна порча данных.

В ППП МК применяется второй способ, основанный на использовании таймера временной метки TS.

Базовый адрес таймера временной метки в линейном адресном пространстве микроконтроллера определен в файле ./psp\_mc/arch/kmk/uke.h.

#define TIMER\_TS\_BASE 0xbb503140

Таймер TS содержит одиннадцать 32-разрядных регистров [5]. Наиболее часто используемые из них представлены в списке ниже.

Control (смещение 0х00) – регистр управления,

SubSecondIncrement (смещение 0х04) – регистр величины инкрементирования,

SystemTimeSeconds (смещение 0х08) – регистр системного времени, секунды (только чтение),

SystemTimeNanoseconds (смещение 0х0с) – регистр системного времени, наносекунды (только чтение),

SecondsUpdate (смещение 0х10) – регистр обновления, секунды,

NanosecondsUpdate (смещение 0х14) – регистр обновления, наносекунды.

Таймер системного времени тактируется внутренней частотой 100 МГц. Каждые 10 наносекунд он увеличивает значение регистра наносекунд SystemTimeNanoseconds на значение, записанное в регистр SubSecondIncrement. По переполнению регистра SystemTimeNanoseconds (каждую секунду) добавляется единица в регистр секунд SystemTimeSeconds.

Необходимо отметить, что таймер системного времени TS может работать, как будильник. Если в регистры TargetTimeSeconds и TargetTimeNanoseconds записать «время побудки», то по достижению его таймер TS сгенерирует прерывание, которое может быть обработано пользовательским программным обеспечением.

***Инициализация таймера TS***

В момент инициализации микроконтроллера исполняется код из файла ./psp\_mc/arch/kmk\_init/kmk\_init.S (на языке Ассемблер). Ниже представлен листинг части этого файла.

Внимание, регистры t7..t9 – это внутренние регистры процессорного ядра микроконтроллера Комдив-МК архитектуры MIPS (всего 32 регистра). Все остальные упоминания регистров относятся к таймеру TS (вернее, к линейному пространству адресов памяти, отраженных на регистры таймера).

/\* initialize timestamp timer \*/

li t9, TIMER\_TS\_BASE /\* Timestamp ctrl register \*/

lb t8, (t9)

andi t8, t8, 0xFE /\* clear TSENA bit \*/

sb t8, (t9) /\* now timer is disabled \*/

li t7, 10

sb t7, 4(t9) /\* Increment 10ns per tick \*/

sw zero, 16(t9) /\* Initial seconds \*/

sw zero, 20(t9) /\* Initial nanoseconds \*/

li t8, 0x205 /\* TSCTRLSSR | TSINIT | TSENA \*/

sw t8, (t9) /\* Timer should be started \*/

В первой инструкции базовый адрес таймера временной метки записывается в регистр процессора t9. Далее в регистр t8 попадает однобайтовое значение из младшего байта регистра Control. Считанное значение побитно умножается на число 0xFE=0b11111110. Таким образом, значение бита TSENA (самый младший бит) обнуляется. Полученный байт записывается обратно по адресу, хранящемуся в регистре t9.

После выполнения четвертой инструкции таймер TS остановлен.

В остановленный таймер можно записать начальное значение и снова стартовать его работу.

В пятой инструкции в регистр t7 записывается число 10. Это число соответствует времени в наносекундах, на которое увеличивается счетчик TS каждый «тик». В шестой строке число 10 попадает в поле SSINC в регистре SubSecondIncrement, адрес которого TIMER\_TS\_BASE + 4.

В седьмой и восьмой строках обнуляются значения начальных секунд и наносекунд в регистрах SecondsUpdate и NanosecondsUpdate по смещениям 16 и 20 относительно базового адреса, соответственно.

Значение 0x205=0b1000000101 – это выставленные нулевой, второй и девятый биты, что соответствует полям разрешения счета TSENA, инициализации системного времени TSINIT и десятичному режиму переполнения наносекунд TSCTRLSSR в регистре Control.

В последней строке это двухбайтовое число 0x205 из регистра t8 процессора записывается в регистр Control таймера TS. Тем самым таймер временной метки снова запускается.

***Функция \_\_micros()***

Макрос \_\_micros определен в файле ./psp\_mc/arch/kmk/kmk\_uke.h.

unsigned long kmk\_micros(void);

#define \_\_micros kmk\_micros

Он ссылается на функцию kmk\_micros(void), выдающую количество микросекунд, прошедших с момента старта микроконтроллера. Формат выдачи – 32-битное целое число. Внимание, переполнение этого счетчика микросекунд наступает примерно через 71 минуту.

Исходный код функции kmk\_micros(void) размещен в файле ./psp\_mc/arch/kmk\_init/kmk\_init.S. Ниже приведен листинг этой функции.

kmk\_micros:

li t9, TIMER\_TS\_BASE + 8

lw t8, (t9) /\* load seconds \*/

lw v1, 4(t9) /\* load nanoseconds \*/

lw v0, (t9) /\* load seconds again - maybe between lw t8 and lw t7

counter was incremented seconds and reset nanoseconds \*/

beq v0, t8, 1f

nop

lw v1, 4(t9) /\* reload nanoseconds. Next inconsistence will occur in

1 sec, so we guarantee not having a one now \*/

1:

/\* Secs in v0 and nanosecs in v1 \*/

li t8, 1000000

mul v0, v0, t8 /\* seconds to microseconds \*/

li t8, 4294968

multu v1, t8

mfhi v1

addu v0, v0, v1

jr ra

nop

По адресу TIMER\_TS\_BASE + 8 (смещение 8) находится регистр секунд системного времени SystemTimeSeconds, а по смещению 12 – регистр наносекунд SystemTimeNanoseconds. Оба регистра доступны только на чтение.

В первой строчке кода в регистр t9 процессора записывается адрес регистра SystemTimeSeconds. Затем в регистры t8 и v1 переносятся значения текущего времени в секундах и наносекундах из регистров SystemTimeSeconds и SystemTimeNanoseconds. Две операции выполняются последовательно, не за один такт процессора. Пока считывалось значение секунд, значение наносекунд могло измениться и переполниться, а значение секунд могло увеличиться на 1. Чтобы удостовериться, что переполнение не случилось, значение секунд повторно считывается в регистр v0 (четвертая строчка).

Далее проверяется, совпадают ли значения регистров v0 и t8. Если они не равны, то в v1 повторно вычитываются наносекунды. Если значения одинаковые, то переход на строку li t8, 1000000.

Начиная с нее, выполняется последовательность арифметических операций быстрого целочисленного деления, описанная в [6].

Функция kmk\_micros(void) возвращает рассчитанное значение микросекунд через регистр v0.

Завершается функция возвратом в место вызова с помощью jr ra и последующего nop. Пустой оператор nop должен быть вставлен после оператора перехода из-за особенностей конвейера процессорного ядра архитектуры MIPS.

***Чтение регистра* SystemTimeNanoseconds *напрямую***

В некоторых случаях для ускорения работы программы можно не пользоваться макросом \_\_micros(), а считывать текущее значение наносекунд напрямую из регистра SystemTimeNanoseconds.

Соответствующий код на языке Си представлен ниже.

unsigned long temp;

temp = \*((volatile unsigned long \*)(TIMER\_TS\_BASE + 12));

В скобках справа рассчитывается адрес регистра SystemTimeNanoseconds – к базовому адресу таймера временной метки TS прибавляется смещение 12. Затем происходит переименовывание указателя: результирующий адрес теперь указывает на целочисленное 32-битное беззнаковое значение. Модификатор volatile запрещает компилятору использовать оптимизацию при работе с памятью. Для того, чтобы «добыть» значения наносекунд, надо обязательно опросить внешнее устройство, привязанное к рассчитанному адресу. Левая звездочка формулирует задачу вернуть значение, взятое по указателю на тип данных unsigned long.

***Функция delay()***

Самый простой (и неэффективный) вариант организации программной задержки в программе – зацикливание и ожидание определенного значения системного таймера. Именно так реализована функция delay(), представленная ниже.

void delay(unsigned long ms) {

unsigned long start = \_\_micros();

while (ms > 0) {

while ( ms > 0 && (\_\_micros() - start) >= 1000) {

ms--;

start += 1000;

}

}

}

Функция delay() принимает на вход целое беззнаковое 32-битное число миллисекунд и уменьшает в цикле оставшееся время ожидания, блокируя дальнейшее выполнение программы до тех пор, пока счетчик ms не дойдет до нуля. Считывание системного времени осуществляется с помощью макроса \_\_micros().

***Задания для самостоятельной работы***

Задания к лабораторной работе подразумевают, что работа выполняется в ОС Windows с предустановленной виртуальной машиной WSL2 с Debian [2] и ППП МК [3-4]. 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 измеряет время, затрачиваемое на однократное считывание временной метки.

Для этого необходимо организовать массив данных достаточно большой длины N, а затем в цикле N раз считать в элементы этого массива системное время двумя способами: с помощью макроса \_\_micros() и прямым чтением регистра SystemTimeNanoseconds. Перед циклом и сразу после цикла также сделать замеры времени. Распечатать на экран полученные временные метки, изучить гранулярность. Проанализировать полученные результаты.

1. Факультативно. Попытаться считать сразу два смежных 32-разрядных регистра SystemTimeSeconds и SystemTimeNanoseconds в одну 64-разрядную целочисленную переменную типа unsigned long long. Скомпилировать проект, найти в полученном дизассемблированном файле \*.dis исходный код этой операции, попытаться его «расшифровать» и показать, что операция чтения – не атомарна. Объяснить, почему так.

Для вывода результата в консоль можно воспользоваться функцией \_\_print\_console(). Для вывода целочисленных 32-битных значений необходимо создать функцию перевода числа в строку, либо воспользоваться сторонней библиотекой, содержащей функцию типа itoa().

***Список литературы***

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

[2] 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

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

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

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

[6] Быстрое целочисленное деление на константу // https://habr.com/ru/articles/147096/