**Лабораторная работа 3**

**Сторожевой таймер WDT**

Мощевикин А.П., ПетрГУ

***Общие сведения о сторожевом таймере WDT в микроконтроллере Комдив-МК***

Микроконтроллер Комдив-МК содержит восемь 64-разрядных таймеров/счетчиков общего назначения и один специализированный – WatchDog Timer (WDT) [1].

Сторожевой таймер WDT используется для перезагрузки микроконтроллера в случае, если «что-то пошло не так».

Обычно встроенное программное обеспечение микроконтроллера строится на основе бесконечного цикла, внутри которого каждую итерацию выполняются некоторые действия: пробежались по всем источникам информации, обработали их, и – снова на начало цикла. В случае, если какое-то внешнее событие (не запланированное программистом) блокирует работу функций в теле цикла, то устройство покажется «зависшим». Примером подобных зависаний может служить некорректная обработка сбоев в передаче данных по сети, неожиданный выход из строя SD-карты в момент чтения/записи, одновременные множественные события, обработка которых взаимно зависит и блокирует друг друга.

Для многих типов бытовых устройств на основе микроконтроллера в случае зависания допустимо перезагрузить систему и возобновить работу, не уделяя времени на разбор причины события и защиту от него. Это можно сделать, используя сторожевой таймер.

Таймер WDT считает вниз (к нулю), начиная с предустановленного значения. Как только счет дойдет до нуля, схемотехника блока WDT сформирует сигнал RESET микроконтроллеру.

Если в каждой итерации главного цикла взводить таймер WDT на достаточно большое время, то пока зависаний нет, все работает корректно. Но как только задержка в цикле превысит время счета таймера WDT, произойдет аппаратная перезагрузка микроконтроллера.

Общая структура исходного кода программы на языке Си выглядит следующим образом.

int main(void) {

init\_board();

while (1) {

update\_WDT();

app\_cycle\_tick();

}

return (1);

}

Базовый адрес контроллера WDT определен в файле ./psp\_mc/arch/kmk/uke.h.

#define TIMER\_WD\_BASE 0xbb503120

Взаимодействие со сторожевым таймером осуществляется через 4 регистра [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.

Для удобства работы имеет смысл создать вспомогательные макроопределения, описывающие внутреннюю структуру этих регистров.

#define WDT\_CTRL\_CNT\_EN 0

#define WDT\_CTRL\_RST\_EN 3

#define WDT\_CTRL\_CLK\_SEL 7

#define WDT\_CTRL\_DIV 16

#define WDT\_STATUS\_ST 0

Первые три описывают номера битов в регистре Control: бита разрешения счета CNT\_EN, бита разрешения формирования сигнала сброса RST\_EN и бита выбора источника тактирования CLK\_SEL (100 МГц или внешний сигнал).

Четвертое определение указывает на смещение поля DIV в регистре Control. Поскольку длина поля DIV составляет 11 битов, оно занимает c 16-го по 26-ой биты в регистре Control. В поле DIV заносится значение предделителя тактовой частоты. Если значение этого поля равно 0x7FF, то таймер WDT не будет использовать предделитель: значение счетчика будет уменьшаться на 1 каждые 10 наносекунд (при тактировании от внутреннего сигнала с частотой 100 МГц).

В последней строке обозначен номер бита ST (состояние) в регистре Status.

Также удобно задать макросами смещения этих регистров.

#define WDT\_CTRL\_REG\_OFFS 0x00

#define WDT\_STATUS\_REG\_OFFS 0x04

#define WDT\_LOAD\_REG\_OFFS 0x08

#define WDT\_READ\_REG\_OFFS 0x10

Для того, чтобы запустить WDT, необходимо выполнить следующую последовательность действий [1].

1. Снять разрешение на счет, записав «0» в CNT\_EN.
2. Проверить состояние бита ST и сбросить его, записав в него значение «1».
3. Установить начальное значение счетчика, записав его в регистр Load.
4. Запустить счетчик, выставив значения в регистре Control: RST\_EN=1, CNT\_EN=1 и, например, DIV=0x7FF, CLK\_SEL=1 (выбрать внутренний источник тактирования с частотой 100 МГц без предделителя).

На языке Си этот фрагмент исходного кода приведен ниже.

unsigned long temp;

\*(volatile unsigned long \*)(TIMER\_WD\_BASE + WDT\_CTRL\_REG\_OFFS) = (0x7FF << WDT\_CTRL\_DIV) | (1 << WDT\_CTRL\_CLK\_SEL) | (1 << WDT\_CTRL\_RST\_EN);

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

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

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

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

\*(volatile unsigned long \*)(TIMER\_WD\_BASE + WDT\_STATUS\_REG\_OFFS) = (1 << WDT\_STATUS\_ST);

\*(volatile unsigned long \*)(TIMER\_WD\_BASE + WDT\_LOAD\_REG\_OFFS) = 100000000; // 1 sec @100 MHz

\*(volatile unsigned long \*)(TIMER\_WD\_BASE + WDT\_CTRL\_REG\_OFFS) = (0x7FF << WDT\_CTRL\_DIV) | (1 << WDT\_CTRL\_CLK\_SEL) | (1 << WDT\_CTRL\_RST\_EN) | (1 << WDT\_CTRL\_CNT\_EN);

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

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

Конструкцию \*(volatile unsigned long \*)(TIMER\_WD\_BASE + WDT\_CTRL\_REG\_OFFS) следует трактовать так. Сначала в правых скобках рассчитывается адрес регистра Control: к базовому адресу 0xbb503120 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 WDT. 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 сторожевого таймера, поскольку этот регистр «закреплен» за этим адресом в линейном пространстве памяти.

Необходимо обратить внимание, что в 64-разрядный регистр Load записывается 32-разрядное значение типа unsigned long (см. предпоследнюю строчку), а не 64-разрядная величина unsigned long long. Поскольку число 100000000 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.

***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 мигнуть три раза по 1 секунде каждый, затем через 5 секунд завершится с исключением.

Структура программы (без бесконечного цикла внутри) представлена ниже.

int main(void) {

init\_board();

for(i=0;i<3;i++) {

. . .

}

delay(5000);

return (1);

}

1. Добавить в программу однократное взведение таймера WDT до цикла for() и вывод значения младшей половины 64-разрядного регистра Read внутри каждой итерации цикла. Поэкспериментировать с длительностью счета таймера WDT (менее секунды, 7 секунд, 10 секунд). Сделать выводы.
2. Изменить значение предделителя DIV в регистре Control, следуя указаниям в [1, раздел 2.3.2], и рассчитать соответствующее начальное значение таймера WDT, записываемое в регистр Load. Убедиться в адекватности результата.
3. Изменить программу, вставив взведение таймера WDT внутрь цикла for(). Поэкспериментировать с длительностью счета таймера WDT (менее секунды, несколько секунд). Сделать выводы.

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