**Лабораторная работа 9**

**Широтно-импульсная модуляция**

Мощевикин А.П., ПетрГУ

***Общие сведения о ШИМ***

Широтно-импульсная модуляция (ШИМ, Pulse Width Modulation, PWM) – техника формирования импульсов с переменной шириной (длительностью), см. рисунок 1. При этом говорят об изменении скважности.

Скважность – это отношение периода следования импульсов (Period) к длительности самого импульса (Width), то есть S = Period / Width. Для сигнала «меандр» скважность равна двум, длительность верхней полки сигнала равна длительности нижней.



Рис. 1 – Изменение скважности при широтно-импульсной модуляции [1].

Вторым, связанным с ШИМ, понятием является duty cycle (рабочий цикл, или коэффициент заполнения) – это величина, обратная скважности. Duty cycle = Width / Period. Обычно выражается в процентах. Например, при коэффициенте заполнения 33% верхняя полка в два раза короче нижней.

На практике ШИМ используется в двух случаях: для управления мощностью и для передачи данных.

*1. Управление мощностью с помощью ШИМ*

Управление мощностью осуществляется методом пульсирующего включения и выключения тока в нагрузке. Такое регулирование подходит только для инерционных процессов. Как правило, частота следования импульсов должна многократно превышать собственные частоты механизмов или устройств. Скажем, для двигателей, вращающихся на частоте 50 Гц, управление должно осуществляться на килогерцах.

То же самое для светодиодов. Человеческий глаз не реагирует на частоты выше 24 Гц. Сигналы с более высокими частотами как бы «размазываются» (усредняются) во времени. Если период следования импульсов установить 1 миллисекунду (частота 1000 Гц), и изменять рабочий цикл, то человеку будет казаться, что светодиод горит постоянно ярко при высоком коэффициенте заполнения и постоянно слабо при низком коэффициенте заполнения.

В этой лабораторной работе поставлена задача научиться управлять яркостью светодиода с помощью широтно-импульсного модулятора, встроенного в микроконтроллер Комдив-МК.

*2. Передача данных с помощью ШИМ*

Кодирование и декодирование цифровых сигналов с широтно-импульсной модуляцией часто используется при передаче данных. Например, при оцифровке аналогового звукового сигнала (при преобразовании его в поток битов). Или при побитной передаче по шине 1-wire: короткий отрицательный импульс означает переданную логическую единицу, а длинный – логический ноль.

***Общие сведения о ШИМ в микроконтроллере Комдив-МК***

Микроконтроллер Комдив-МК содержит восемь 64-разрядных таймеров/счетчиков общего назначения [2]. В лабораторной работе №5 «Использование таймера TIMER0 в режиме прерываний» описан принцип их работы.

Каждый из таймеров общего назначения может работать в режиме широтно-импульсного модулятора, т.е. формировать прямоугольный сигнал переменной скважности на выводах микроконтроллера.

Реализуется это следующим образом. У каждого таймера есть два регистра: регистр Load, куда заносится длительность цикла, и регистр PWM, в котором указывается длительность верхней полки импульса.

При работе таймера в бесконечном режиме во время старта очередного цикла формируется передний фронт импульса на выводе Output (рисунок 1), а также считывается значение из регистра PWM. Счетчик Count начинает увеличивать свое значение каждый тик таймера. Как только значение счетчика Count доходит до значения PWM (на рисунке 1 – событие Match при значении счетчика, равное 2), формируется задний фронт импульса. При достижении счетчиком значения Load происходит событие Overflow, и таймер перезапускается.

Так устроено, что в микроконтроллере Комдив-МК выходы Output ШИМ-генераторов могут быть выведены только на линии порта A (линии GPIO\_A0..GPIO\_A7). Т.е. ШИМ-выход таймера TIMER0 может быть выведен только на линию GPIO\_A0, ШИМ-выход таймера TIMER1 – только на линию GPIO\_A0 и т.д.

Для перевода линии порта ввода-вывода необходимо присвоить ей альтернативное назначение, см. таблицу «2.7.1 Мультиплицирование выводов на кристалле» [3]. Для порта A альтернативное назначение в качестве ШИМ указано под номером 3. Поэтому в коде функции инициализации проекта init\_board(void) необходимо провести соответствующую настройку для выбранной линии порта GPIO\_A.

// PWM0 <–> GPIO\_A0

GPIO\_SetPinMode(GPIOA\_BASE, 0, GPIO\_MODE\_ALT3);

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

Взаимодействие с таймерами осуществляется через 5 регистров (в скобках указано смещение относительно базового адреса каждого таймера) [2]:

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.

Для выполнения этой лабораторной работы имеет смысл дополнительно определить макросы, задающие 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 // предделитель

#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.

Для того, чтобы запустить таймер-счетчик TIMER0 в циклическом режиме с периодом 1 миллисекунда (счет на частоте 100 МГц без предделителя) с ШИМ (duty cycle = 50%), необходимо выполнить следующую последовательность действий [2].

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. Установить начальное значение счетчика 50000, записав его в регистр PWM. Значение 50000 соответствует длительности верхней полки ШИМ-сигнала 500 мкс для тактовой частоты 100 МГц.
5. Запустить счетчик, выставив значения в регистре Control: DIV=0x7FF, CLK\_SEL=1, INT\_EN=1, MODE=1, CNT\_EN=1, PWM\_EN=1. Значение полярности OUT\_POL ШИМ-сигнала можно не устанавливать.

На языке Си этот фрагмент исходного кода приведен ниже.

\*(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);

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

 (1 << TIMER\_STATUS\_ST);

\*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_LOAD\_REG\_OFFS) = 100000; // \* 10 ns

\*(volatile unsigned long \*)(TIMER0\_BASE + TIMER\_PWM\_REG\_OFFS) = 50000;

\*(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) | \

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

Необходимо обратить внимание (см. третью операцию), 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.

В последней операции помимо старта таймера также разрешается работа широтно-импульсного модулятора (выставляется бит PWM\_EN).

***Подготовка к выполнению лабораторной работы***

Для выполнения лабораторной работы необходимо собрать на макетной плате цепь из резистора и светодиода и подключить ее к микроконтроллеру в соответствии с рисунком 2.

 

Рис. 2 – Подключение компонентов для лабораторной работы.

Токоограничивающий резистор можно выбрать любой в диапазоне от 200 Ом до 1 кОм. От его номинала зависит яркость свечения светодиода. Чем меньше номинал резистора, тем выше ток протекает через светодиод и тем ярче он светится.

Цепь резистор-светодиод одним концом подключается к выбранной линии порта А, а другим – к земле.

***Задание для самостоятельной работы***

Используя исходный код лабораторной работы 5 для проекта на языке Си, создать программу, которая заставляет светодиод циклически с периодом 1 секунда плавно разгораться и плавно гаснуть (реализовать функцию ШИМ-диммирования или «дыхания», breathing).

Структура программы представлена ниже.

int main(void) {

 init\_board();

 while (1) {

 . . .

 }

 return 1; // unreacheable

}

Программа не должна использовать прерывания.

Внутри основного цикла необходимо периодически обновлять значение регистра PWM. Например, с использованием пауз delay() между обновлениями.

Необходимо отметить, что такой способ управления ШИМ без прерываний в режиме реального времени – достаточно «грязный». Правильнее было бы дожидаться конца каждого цикла (события Overflow), далее в прерывании останавливать таймер, устанавливать новое значение PWM, и только потом снова запускать таймер.

Если записываемое в регистр PWM значение имеет формат unsigned long long (64-битное число), то запись в этот регистр будет проходить в две стадии: сначала в контроллер таймера из процессора произойдет передача одной половины числа (первые 4 байта), затем – второй. И если в момент, когда первая половина уже записана в контроллер, а вторая – еще нет, случайно произойдет переполнение счетчика, может возникнуть сложно отлавливаемая в коде ошибка.

Однако, если значение длительности верхней полки сигнала не превышает формат 32-битного числа, то запись значения в регистр PWM происходит атомарно, следовательно, нарушений в работе не будет.

Разработчики платы БАГЕТ-ПЛК1-01 [4] рекомендуют использовать среду разработки и структуру проекта, описанные в [5] и [6, раздел 3].

Задания к лабораторной работе подразумевают, что работа выполняется в ОС Windows с предустановленной виртуальной машиной WSL2 с Debian [5] и ППП МК [6-7]. Проекты на языке Cи следует создавать в папке ./psp\_mc/apps/ .

Для формирования импульса можно использовать функцию delay\_us(), для замера времени работы –\_\_micros(), для вывода результата – \_\_print\_console(), для форматирования строки вывода – itoa() и intToHex().

***Список литературы***

[1] Pulse-width Modulation (PWM) Timers in Microcontrollers // https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-pwm-timers/

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

[3] Микросхема интегральная К5500ВК018. Контроллер GPIO. ЮКСУ.431295.019Д4.19 // К5500ВК018\_d4\_19\_GPIO\_v.2.4.2.doc

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

[5] 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

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

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