

Лабораторная работа №5

Измерение частоты сигнала

Цель работы: научиться измерять частоту, используя таймеры микроконтроллера и вывести значение на экран.

Оборудование и программное обеспечение: плата UDK32F107V, среда разработки Cube IDE, генератор сигналов XR2206, беспаячная макетная плата, резистор 1 кОм – 2шт, соединительные провода.

Теоретическая информация

Частотомер

Частотомер — это электроизмерительный прибор, предназначенный для измерения частоты или периода колебаний электрического сигнала.

- Классифицируют частотомеры, в основном:
- По методу измерения — приборы непосредственной оценки (аналоговые) и приборы сравнения (резонансные, гетеродинные, электронно-счетные).
- По физическому смыслу измеряемой величины — для измерения частоты синусоидальных колебаний (аналоговые), измерения частот гармонических составляющих (гетеродинные, резонансные, вибрационные) и измерения частоты дискретных событий (электронно-счетные, конденсаторные).
- По исполнению — щитовые, переносные и стационарные.

Наибольшее распространение получили очень удобные и точные универсальные частотомеры прямого отсчета — цифровые частотомеры. Принцип действия, которых заключается в подсчете количества импульсов, которые поступают от входного формирователя за строго определенный интервал времени. Такой прибор позволяют измерять не только частоту, но и временные интервалы, и количество импульсов.

Цифровые методы измерения частоты весьма просты в реализации и обеспечивают высокую точность. Особенно следует выделить метод

обратного счёта - он не только точный (причём точность сохраняется во всём диапазоне измеряемых частот), но и способен обеспечить высокую скорость измерения. Метод заключается в следующем. Формируется интервал времени, состоящий из целого количества m периодов исследуемого сигнала, и подсчитывается количество импульсов n эталонного генератора с частотой F_0 за этот интервал времени. Величина m выбирается таким образом, чтобы время счёта было примерно равно заданному желаемому интервалу измерения (подобрать величину можно прямо в процессе измерения). По результатам счёта вычисляем среднее значение периода сигнала (или частоты сигнала) за интервал измерения.

Функциональный генератор

Аналоговый функциональный генератор (рис. 1) синусоидальных, треугольных и прямоугольных сигналов в диапазоне частот от 1Гц до 1МГц для применения в радиолюбительской практике.

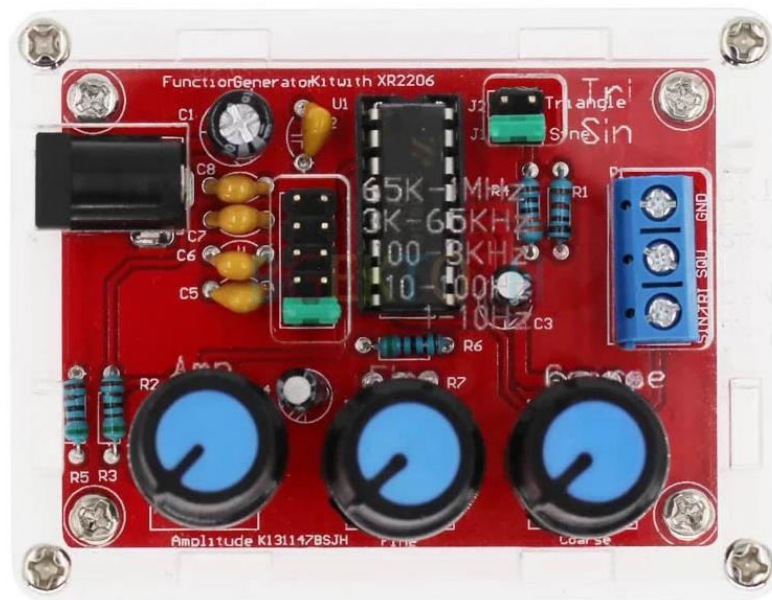


Рис.1. Функциональный генератор

Основой схемы генератора сигналов является чип XR2206. Напряжение питания — 9...15В, амплитуда выходного сигнала — до 3В (синус/треугольник) и до 8В — меандр.

Настройка частоты выполняется двумя резисторами — «грубая» — Coarse и «точная» — Fine настройка, а также джампером выбора диапазона частот. Всего таких диапазонов пять:

1. 1-10 Гц
2. 10-100 Гц
3. 100 Гц — 3 кГц
4. 3 кГц — 65 кГц
5. 65 кГц — 1 МГц

Выходной сигнал снимается с клеммника, назначение его выводов:

1. GND — земля, т.е. «общий» провод
2. SQU — выход прямоугольных импульсов
3. SIN / Tri — синусоидальные или треугольные импульсы. Выбор

типа сигнала на этом пине осуществляется джампером Tri / Sin

Амплитуда выходного сигнала для синусоиды и треугольника регулируется переменным резистором Amp. Амплитуда меандра не регулируется.

Технические характеристики показаны в таблице 1.

Таблица 1. Технические характеристики генератора сигналов

Диапазон генерируемых частот	1Гц ... 1МГц
Амплитуда синус / треугольник	к 0...3 В
Амплитуда прямоугольник	8 В
Напряжение питания	9...15 В
Выходное сопротивление	600 Ом
Размер по корпусу	71x54x17 мм

Пример программы

Для начала выполнения работы подключим функциональный генератор к плате UDK32F107 по схеме, как показано на Рис. 2

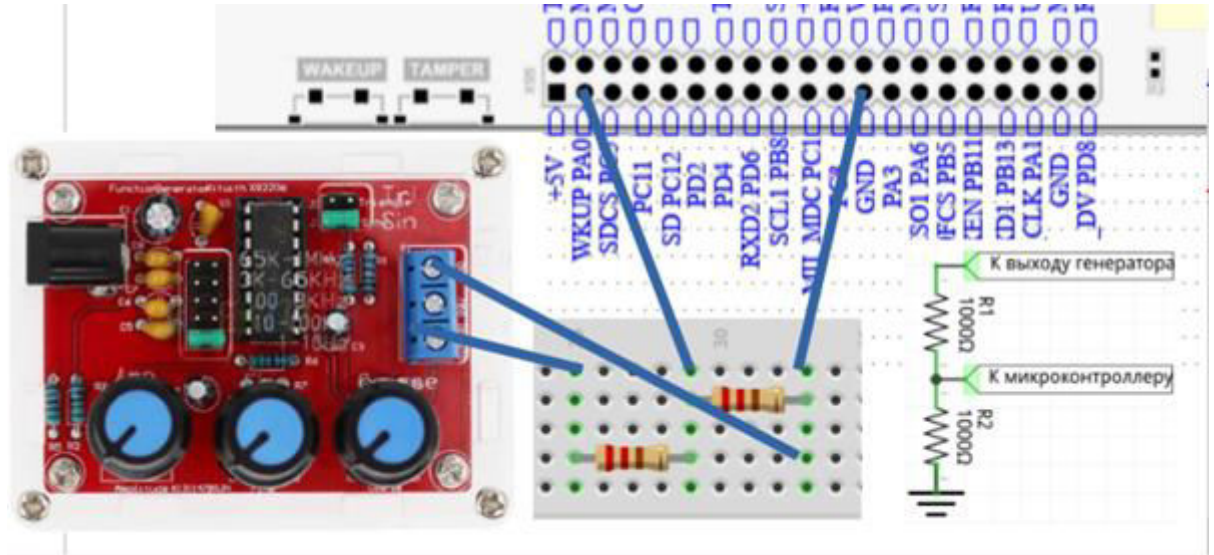


Рис. 2. Схема подключения генератора сигналов

Настраиваем тактирование микроконтроллера (рис. 3).

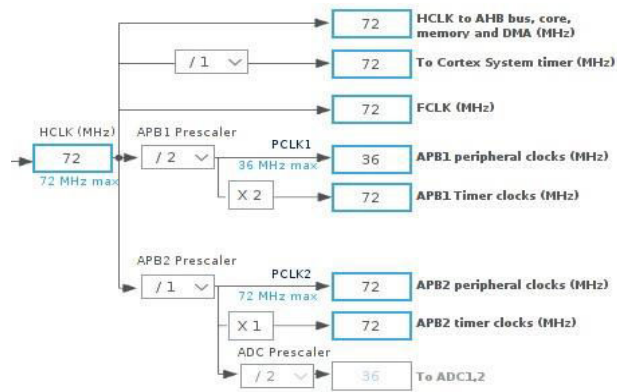


Рис. 3. Настройка тактирования микроконтроллера

Активируем пин RCC_MCO (Master Clock Output), он будет выдавать «эталонную» частоту для проверки нашего частотомера (рис. 4). Ставим галочку Master Clock Output и указываем максимальную скорость (Maximum output speed) для пина PA8 (RCC MCO).

RCC Mode and Configuration

Mode

High Speed Clock (HSE) Crystal/Ceramic Resonator

Low Speed Clock (LSE) Disable

☒ Master Clock Output

Configuration

Reset Configuration

☒ User Constants
 ☒ NVIC Settings
 ☒ GPIO Settings

☒ Parameter Settings

Search Signals

☐ Show only Modified Pins

Pin ...	Signal o...	GPIO ou...	GPIO m...	GPIO Pu...	Maximu...	User La...	Modified
PA8	RCC_M...	n/a	Alterna...	n/a	High		<input checked="" type="checkbox"/>
PD0-OS...	RCC_O...	n/a	n/a	n/a	n/a		<input type="checkbox"/>
PD1-OS...	RCC_O...	n/a	n/a	n/a	n/a		<input type="checkbox"/>

PA8 Configuration :

GPIO mode Alternate Function Push Pull

Maximum output speed High

User Label

Рис. 4. Настройка порта PA8 на выход сигнала тактового генератора

Источником для выхода MCO указываем HSE (рис. 5).

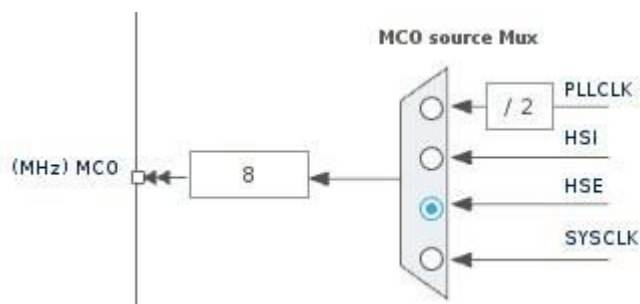
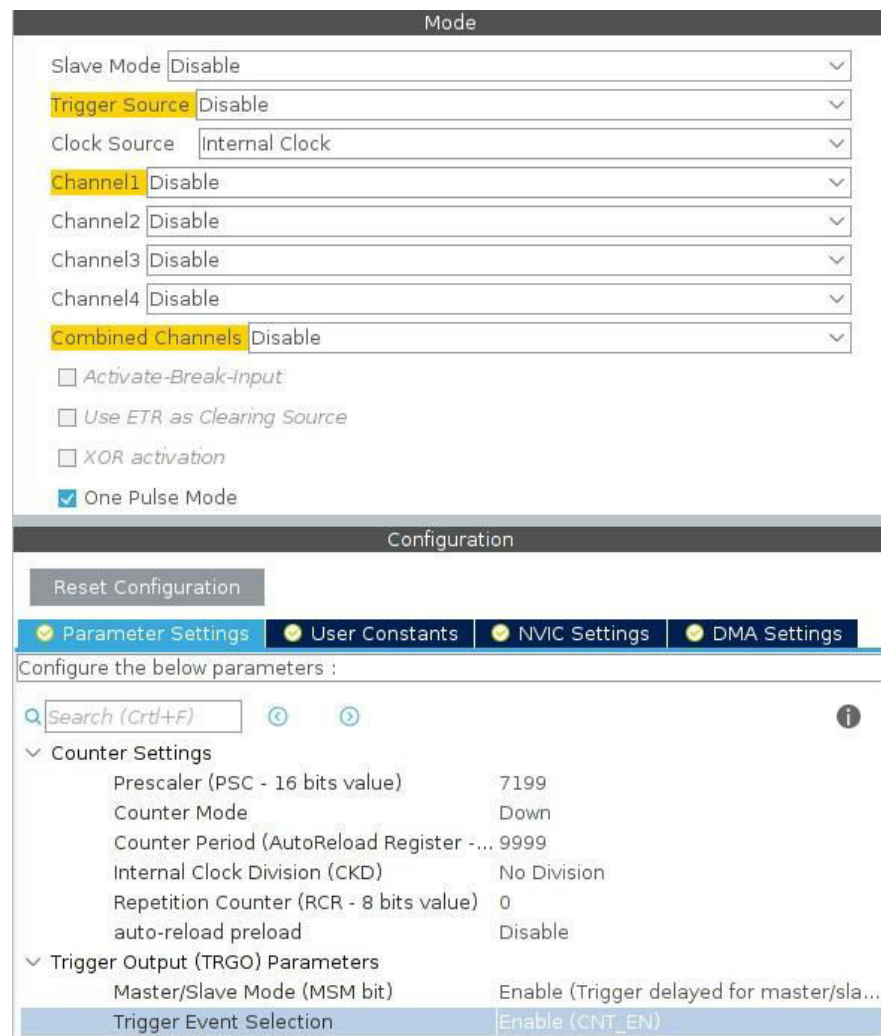


Рис. 5. Выбор источник тактового сигнала для порта PA8



Mode

Slave Mode: Disable

Trigger Source: Disable

Clock Source: Internal Clock

Channel1: Disable

Channel2: Disable

Channel3: Disable

Channel4: Disable

Combined Channels: Disable

☐ Activate-Break-Input

☐ Use ETR as Clearing Source

☐ XOR activation

☒ One Pulse Mode

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits value)	7199
Counter Mode	Down
Counter Period (AutoReload Register - ...)	9999
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

Master/Slave Mode (MSM bit)	Enable (Trigger delayed for master/sla...
Trigger Event Selection	Enable (CNT_EN)

Рис. 6. Настраиваем таймер №1

Настраиваем таймеры 1, 2 и 3, как показано на рис. 6, 7 и 8. Таймер №1 будет отмерять интервал времени равный одной секунде. Таймер №2 тактируется от измеряемой частоты через ETR-пин, и тем самым его счётчик CNT (Counter Period) будет выступать в роли счётчика импульсов. Поскольку таймеры у stm32 16-ти битные, то таймер №2 сможет насчитать не более 65535 импульсов, то есть можно измерить частоту не более 65.535 КГц. Чтобы преодолеть этот барьер нужно использовать таймер №3, который будет отсчитывать количество переполнений таймера №2. То есть, таймеры №2 и №3 будут работать каскадно, как один 32-х битный.

Таймер будет переполняться раз в секунду и генерировать прерывание, включите его на вкладке NVIC Settings ⇒ TIM1 update interrupt.

- One Pulse Mode — таймер будет отрабатывать один раз (один цикл переполнения), генерировать прерывание и останавливаться, перезапускать его будем «вручную» в колбеке.
- Trigger Event Selection Enable(CNT_EN) — триггерный сигнал (HIGH), подаётся с момента старта таймера и удерживается до его остановки, после остановки таймера сигнал снимается (переключается в LOW).

При старте, таймер №1 будет аппаратно подавать этот сигнал на таймер №2 и тем самым запускать его. При остановке таймера №1 этот сигнал будет снят и таймер №2 остановится. То есть, мы «вручную» запустили таймер №1 (пошёл отсчёт секунды), а он в свою очередь «толкает» таймер №2, который начинает тактироваться от измеряемой частоты и складывать поступающие импульсы в свой счётчик. Таким образом у нас получается следующее: оба таймера, №1 (отмеряющий секунднй интервал) и №2 (считающий импульсы) стартанут чётко одновременно, а когда таймер №1 отмерит секунду и остановится, то и таймер №2 тоже остановится (перестанет считать импульсы) — в счётчике таймера №2 будет лежать количество импульсов входящей частоты полученные за одну секунду (при условии что частота была не больше 65.535 КГц, если частота выше, то в дело вступит таймер №3, который будет подсчитывать количество переполнений таймера №2).

Master/Slave Mode (MSM bit) Enable... — при активации этого пункта, таймер №1 перед тем как запустится, сделает небольшую паузу чтоб сигнал для запуска таймера №2 успел дойти.

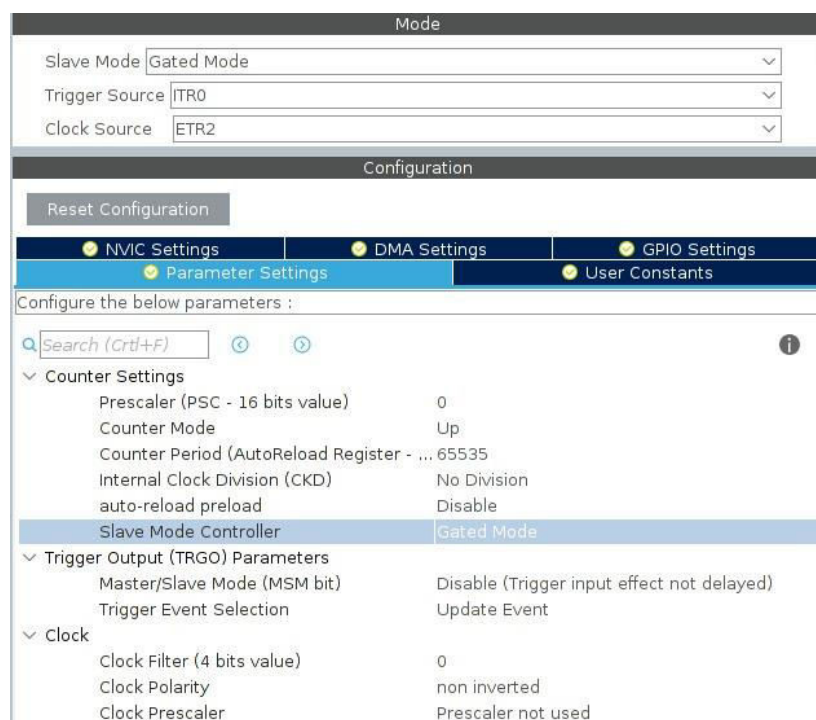


Рис. 7. Настраиваем таймер №2

Slave Mode \Rightarrow Gate Mode — таймер будет работать пока подаётся триггерный сигнал от таймера №1.

Trigger Source \Rightarrow ITR0 — триггерный сигнал посылает таймер №1.

Clock Source \Rightarrow ETR2 — источником тактирования будет частота поступающая на пин TIM2_ETR (PA0). Это та самая частота, которую мы хотим измерить.

Counter Period — значение переполнения устанавливаем максимальное — 65535.

Trigger Event Selection \Rightarrow Update Event — настраиваем исходящий триггерный сигнал, который будет посылаться таймеру №3 в момент переполнения. То есть, таймер №2 переполнился, обнулится, послал импульс таймеру №3 и продолжил считать. Таким образом в счётчике таймера №3 будет лежать количество переполнений таймера №2.

Тут есть ещё один важный для нас пункт — Clock Prescaler, но к нему вернёмся позже.

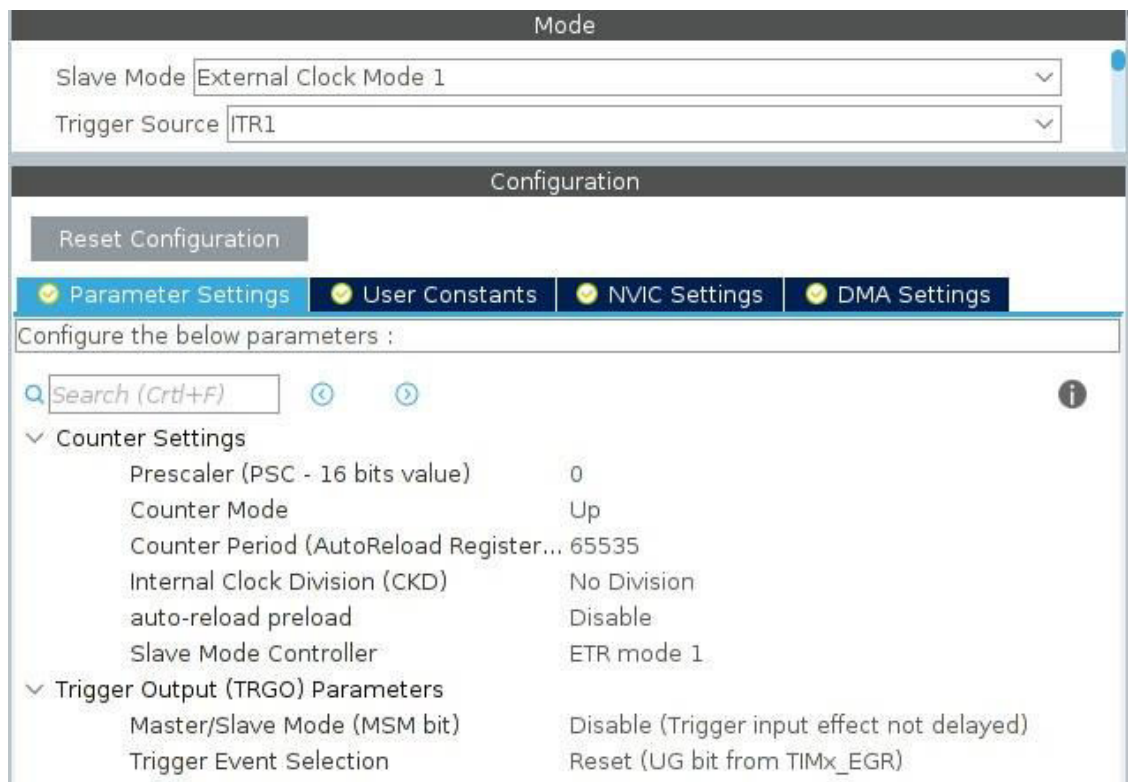


Рис. 8. Настраиваем таймер №3

Slave Mode External Clock Mode 1 — здесь мы указываем что таймер будет тактироваться какими-то внешними (по отношению к таймеру) сигналами.

Trigger Source ITR1 — а здесь говорим что эти тактирующие сигналы будут поступать от таймера №2.

Counter Period — переполнение опять же устанавливаем максимальное — 65535.

Для второго и третьего таймеров никаких прерываний включать не нужно.

В итоге у нас получается следующая схема работы: стартуем таймер №1, он в свою очередь тут же запускает таймер №2, который начинает складывать в свой счётчик импульсы измеряемой частоты. Как только таймер №2 переполняется (при условии что поступающая частота выше 65.535 КГц, если ниже то всё уместится в один цикл), он посылает сигнал таймеру №3, таймер №3 делает один «тик» и записывает в свой счётчик единичку. После переполнения таймер №2 продолжает считать, вновь переполняется и вновь

посылает сигнал таймеру №3, таймер №3 делает ещё один «тик» и записывает в свой счётчик ещё одну единичку. Так будет продолжаться до тех пор, пока не истечёт секунда и таймер №1 не остановится сгенерировав прерывание. При остановке таймера №1 автоматически остановится таймер №2.

Если предположить что мы измеряем частоту 200000 Гц (200КГц), то в идеале получится так: за одну секунду таймер №2 успеет переполниться три раза — $65535 * 3 = 196605$, а после третьего переполнения в его счётчик запишется ещё 3395. По истечении секунды таймер №1 сгенерирует прерывание, в колбеке которого нам остаётся взять значение которое лежит в счётчике таймера №2 и сложить его со значением в счётчике таймера №3 умноженное на значение переполнения, то есть так — $3395 + (3 * 65535) = 200000$ Гц.

Теперь напишем код и проверим работоспособность.

Перед бесконечным циклом запускаем таймер №1 в режиме прерывания, и активируем таймеры №2 и №3

```
/* USER CODE BEGIN Includes */
#include "ssd1289.h"
#include "stdio.h"
/* USER CODE END Includes */

/* USER CODE BEGIN PV */
char str[96] = {0,};
uint32_t freq;
int updateValue;
/* USER CODE END PV */

/* USER CODE BEGIN 0 */
int _write(int32_t file, uint8_t *ptr, int16_t len){
    /* Implement your write code here, this is used by puts and printf for
example */
```

```

int i=0;
for(i=0 ; i<len ; i++)
ITM_SendChar((*ptr++));
return len;
}
/* USER CODE END 0 */

/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim1);
HAL_TIM_Base_Start(&htim2);
HAL_TIM_Base_Start(&htim3);
LCD_Init(); //инициализация дисплея
LCD_FillScreen(BLACK); //делаем заливку дисплея чёрным фоном
/* USER CODE END 2 */

while (1) {
    if (updateValue){
        LCD_WriteString_8x16(80,220,          "STM32          Frequency
meter",WHITE,BLACK); //пишем текст
        sprintf(str, "Freq: %.3f MHz    ", (float)freq / 1000000.0);
        LCD_WriteString_8x16(1,180, str, WHITE, BLACK); //пишем текст
        sprintf(str, "Freq: %.3f KHz    ", (float)freq / 1000.0);
        LCD_WriteString_8x16(1,160, str, WHITE, BLACK); //пишем текст
        sprintf(str, "Freq: %lu Hz      ", freq);
        LCD_WriteString_8x16(1,140, str, WHITE, BLACK); //пишем текст
    }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 4 */

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if(htim == &htim1) {
        updateValue = 0; //
        freq = TIM2->CNT + (TIM3->CNT << 16); // это вариант на регистрах,

```

предыдущие четыре строчки можно закомментировать

```
printf("FREQUENCY: %.3f MHz | %.3f KHz | %lu Hz\n-----  
\n", (float)freq / 1000000.0, (float)freq / 1000.0, freq);  
HAL_TIM_Base_Stop_IT(&htim1);  
__HAL_TIM_SET_COUNTER(&htim2, 0x0000);  
__HAL_TIM_SET_COUNTER(&htim3, 0x0000);  
HAL_TIM_Base_Start_IT(&htim1);  
updateValue = 1;  
}  
}  
/* USER CODE END 4 */
```

Результатом выполнения работы должны получить такую картинку. Следует также отметить, что микроконтроллер выдаёт сигнал частотой 25 МГц на GPIOA_8. Информация о частоте будет выводиться на дисплей и в монитор отладки SWV Data Console.

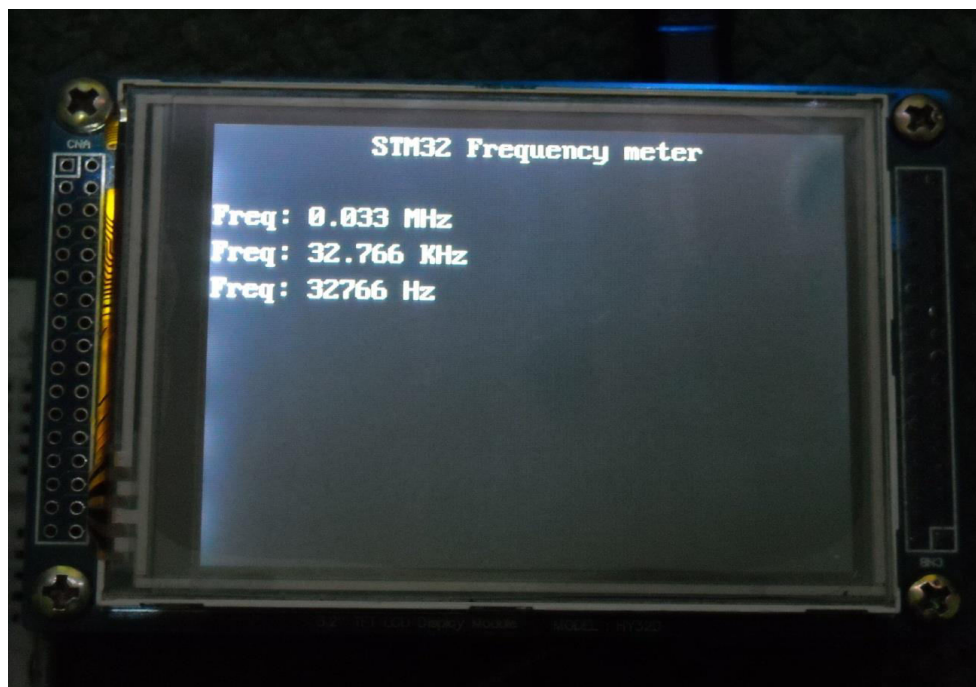


Рис. 9. Работа частотомера

Ход работы

1. На основе кода примера приложения создать свой проект в среде разработки и проверить его работоспособность.
2. Ознакомиться с работой таймеров.
3. Реализовать необходимую функциональность.
4. Запрограммировать плату и продемонстрировать работу программы.