



МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт компьютерных технологий
и информационной безопасности

Методические указания
к выполнению лабораторной работы № 2
«Исследование работы встроенной периферии микроконтроллера»

по курсу
Введение в инженерную деятельность

Составил:
Скляров С. А.

1 Методические указание к выполнению лабораторной работы

В данном методическом пособии рассматривается подключение, настройка и основные возможности встроенной в микроконтроллер периферии.

1.1 Цель работы

Целью работы является изучение встроенной базовой периферии МК, её режимов работы и параметров функционирования.

Задачами работы являются:

- 1) Изучение аналогово-цифрового преобразователя и считывание показаний аналоговых устройств;
- 2) Настройка и управление цифро-аналоговым преобразователем, генерация аналоговых сигналов;
- 3) Использование аппаратных таймеров для генерация широтно-импульсной модуляции;
- 4) Управление системой прерывания микроконтроллера и обработка внешних событий.

1.2 Аналого-цифровой преобразователь

Для получения сигналов из аналоговых устройств используется встроенный в контроллер аналого-цифровой преобразователь. Для активации АЦП (ADC) необходимо установить соответствующий бит в регистре RCC:

```
RCC->APB2ENR |= RCC_APB2ENR_ADC1EN; // ADC задействован
```

Подготавливаем контакт контроллера **PC3** для получения сигнала:

```
RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN; // Порт C задействован  
GPIOC->MODER &= ~GPIO_MODER_MODER3; // Сброс режима для PC3  
GPIOC->MODER |= GPIO_MODER_MODER3; // Аналоговый вход для PC3
```

Далее необходимо разрешить работу АЦП командой:

```
ADC1->CR2 = ADC_CR2_ADON; // ADC активен
```

Выбираем **13** канал для получения аналогового сигнала из **PC3**:

```
ADC1->SQR3 = 13; // Выбран 13 канал
```

Ожидаем когда установленный канал будет готов к преобразованию:

```
for(int a=0; a<20; a++) { } // Ожидать больше 20 тактов
```

Запускаем однократное преобразование:

```
ADC1->CR2 |= ADC_CR2_SWSTART; // Начать преобразование
```

Ожидаем окончание операции преобразования:

```
while(!(ADC1->SR & ADC_SR_EOC)) { } // Ждать установки бита конца операции
```

Считать полученные данные в переменную:

```
int data = ADC1->DR; // Получение результата преобразования
```

Для считывания данных из других устройств нужно изменить настройки согласно таблице 1.

Устройство	Порт	Вывод	Канал
Джойстик X	GPIOC	0	10
Джойстик Y	GPIOC	1	11
Потенциометр 2 (POT2)	GPIOC	2	12
Потенциометр 1 (POT1)	GPIOC	3	13

Таблица 1 – линии АЦП.

Для циклического преобразования нужно повторить операцию преобразования, ожидать её окончание и считать данные.

1.3 Цифро-аналоговый преобразователь

Для генерации аналогового сигнала микроконтроллером используется цифро-аналоговый преобразователь. Для активации ЦАП (DAC) необходимо установить соответствующий бит в регистре RCC:

```
RCC->APB1ENR |= RCC_APB1ENR_DACEN; // DAC задействован
```

Подготавливаем контакт контроллера PA4 для генерации сигнала:

```
RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; // Порт A задействован
```

```
GPIOA->MODER &= ~GPIO_MODER_MODER4; // Сброс режима
```

Далее необходимо разрешить работу DAC командой:

```
DAC->CR |= DAC_CR_EN1; // DAC с каналом 1 активен
```

Задаём значение на выходе активного канала:

```
DAC->DHR12R1 = 500; // Задать генерируемую амплитуду
```

В 12 битном режиме значение амплитуды 0-4096 (0-100%) что на выходе означает 0V-3.3V. При значении амплитуды равной 500 на выходе получим $3.3 * 500 / 4096 = 0.4V$.

1.4 Таймеры и генератор ШИМ

Микроконтроллер содержит таймер для точного отсчета интервалов времени. Они используются для генерации сигналов, вызовов прерываний, синхронизации периферии и т. д. Для активации таймера (TIM) необходимо установить соответствующий бит в регистре RCC:

```
RCC->APB1ENR |= RCC_APB1ENR_TIM4EN; // Таймер 4 задействован (APB1 x2 = 84МГц)
```

Для удобства создадим функцию выбора альтернативного режима:

```
void SetAltFunc(GPIO_TypeDef* Port, int Channel, int AF)
{
    Port->MODER &= ~(3<<(2*Channel)); // Сброс режима
    Port->MODER |= 2<<(2*Channel); // Установка альт. Режима

    if(Channel<8) // Выбор регистра зависит от номера контакта
    {
        Port->AFR[0] &= ~(15<<4*Channel); // Сброс альт. функции
        Port->AFR[0] |= AF<<(4*Channel); // Установка альт. функции
    }
    else
    {
        Port->AFR[1] &= ~(15<<4*(Channel-8)); // Сброс альт. функции
        Port->AFR[1] |= AF<<(4*(Channel-8)); // Установка альт. функции
    }
}
```

Альтернативный режим позволяет передать управление контактом микроконтроллера периферийному устройству, в нашем случае таймер TIM4 получает контроль над PD12, PD13 и PD14.

Подготавливаем контакты контроллера для генерации сигнала:

```
RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN; // Порт D задействован
SetAltFunc(GPIOD, 12, 2); // Установка альт. режима AF2 для TIM4 CH1 (PD12)
SetAltFunc(GPIOD, 13, 2); // Установка альт. режима AF2 для TIM4 CH2 (PD13)
SetAltFunc(GPIOD, 14, 2); // Установка альт. режима AF2 для TIM4 CH3 (PD14)
```

Настраиваем параметр таймера для генерации ШИМ импульса при сравнении регистра CCR:

Задаём диапазон сравнения 1000 минус единица:

```
TIM4->ARR = 1000 - 1; // Диапазон сравнения 1000
```

Устанавливаем делитель с учетом шины APB1(84000000) диапазона (1000) и желаемой частотой 2000Гц минус единица:

```
TIM4->PSC = (84000000/1000/2000)-1; // Задан делитель
```

Выбираем PWM режим 1 и разрешаем предзагрузку регистра CCR:

```
TIM4->CCMR1 = TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1PE; // Режим CCR1
TIM4->CCMR1 |= TIM_CCMR1_OC2M_1 | TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2PE; // Режим CCR2
TIM4->CCMR2 = TIM_CCMR2_OC3M_1 | TIM_CCMR2_OC3M_2 | TIM_CCMR2_OC3PE; // Режим CCR3
```

Разрешаем генерацию выходного сигнала:

```
TIM4->CCER = TIM_CCER_CC1E; // Выход PD12 активен
TIM4->CCER |= TIM_CCER_CC2E; // Выход PD13 активен
TIM4->CCER |= TIM_CCER_CC3E; // Выход PD14 активен
```

Запускаем таймер:

```
TIM4->CR1 = TIM_CR1_CEN; // Таймер запущен
```

Задаём скважность ШИМ (100% * Значение / Диапазон сравнения):

```
TIM4->CCR1 = 500; // Скважность PD12 = 100*500/1000 = 50% (RED)
TIM4->CCR2 = 250; // Скважность PD13 = 100*500/1000 = 25% (GREEN)
TIM4->CCR3 = 750; // Скважность PD14 = 100*500/1000 = 75% (BLUE)
```

1.5 Система прерываний

Прерывания в микроконтроллере служат для обработки внутренних и внешних событий. Создадим прерывание при нажатии кнопки, задействовав контакт PE4:

```
RCC->AHB1ENR |= RCC_AHB1ENR_GPIOEEN; // Порт E задействован
GPIOE->MODER &= ~GPIO_MODER_MODER4; // Сброс режима (режим входа) PE4
```

Создаём функцию для обработки прерывания:

```
extern "C" void EXTI4_IRQHandler() // Название функции для EXTI4
{
    EXTI->PR = 1<<4; // Снять бит активности прерывания (прерывание 4 обработано)
    if (GPIOE->IDR & (1<<4)) // Прочитать значение входящего сигнала в PE4
    {
        //... // Пользовательский код при появлении сигнала (кнопка нажата)
    }
    else
    {
        //... // Пользовательский код при исчезновении сигнала (кнопка отпущена)
    }
}
```

Разрешаем работу системного конфигурируемого контроллера:

```
RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN; // Задействован SYSCFG
```

Пишем функцию для выбора порта и канала:

```
enum PORT {A, B, C, D, E, F, G, H, I}; // Перечисление доступных портов

void SetEXTI(PORT Port, int Channel, bool Rise, bool Fall)
```

```

{
    SYSCFG->EXTICR[Channel/4] &= ~(15<<(4*(Channel%4))); // Сбросить порт
    SYSCFG->EXTICR[Channel/4] |= Port<<(4*(Channel%4)); // Выбрать порт

    EXTI->IMR |= 1<<Channel; // Прерывание выбрано

    if(Rise) EXTI->RTSR |= 1<<Channel; // Ловить повышение напряжения
    else EXTI->RTSR &= ~(1<<Channel); // Не ловить повышение напряжения

    if(Fall) EXTI->FTSR |= 1<<Channel; // Ловить падение напряжения
    else EXTI->FTSR &= ~(1<<Channel); // Не ловить падение напряжения
}

```

Выбрать порт и канал для прерывания:

```
SetEXTI(PORT::E, 4, true, true); // Прерывание PE4 при появлении и исчезновении сигнала
```

Установить приоритет прерывания и разрешить его обработку:

```

NVIC_SetPriority(EXTI4_IRQn, 0); // Высший приоритет прерывания
NVIC_EnableIRQ(EXTI4_IRQn); // Прерывание активировано

```

Теперь при появлении или падении напряжения на **PE4** будет вызываться прерывание и исполняться код в функции «EXTI4_IRQHandler».

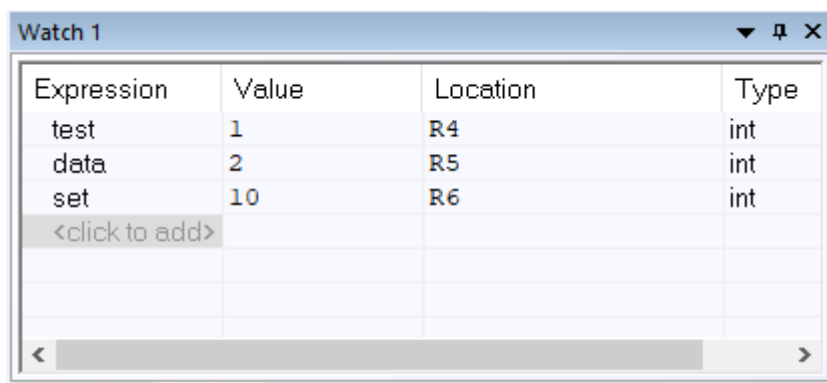
Для обработки прерываний на линиях **0 – 15** используются названия функций и номера прерывания из таблицы 2. Регистр «EXTI->PR» содержит активные биты, соответствующие номерам сработавших внешних прерываний. В нём, чтобы прерывание считалось обработанным, необходимо в соответствующий бит записать «1», иначе вызов прерывания заиклится.

Номер линии	Номер прерывания	Название обработчика
0	EXTI0_IRQn	EXTI0_IRQHandler
1	EXTI1_IRQn	EXTI1_IRQHandler
2	EXTI2_IRQn	EXTI2_IRQHandler
3	EXTI3_IRQn	EXTI3_IRQHandler
4	EXTI4_IRQn	EXTI4_IRQHandler
5-9	EXTI9_5_IRQn	EXTI9_5_IRQHandler
10-15	EXTI15_10_IRQn	EXTI15_10_IRQHandler

Таблица 2 – линии внешних прерываний.

1.6 Расширенная отладка

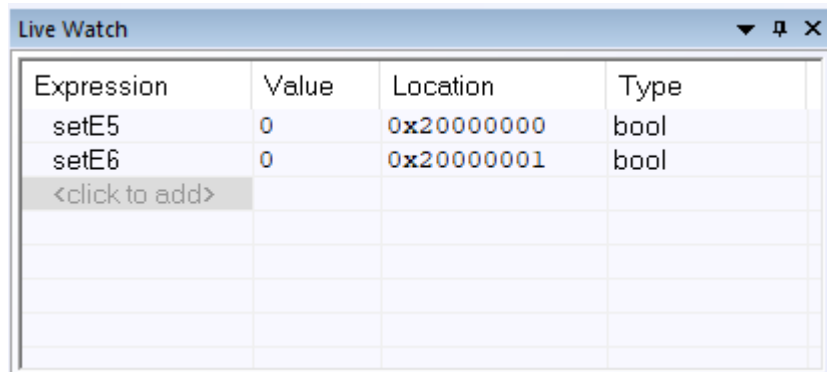
Для считывания текущих значений и получения мгновенных показаний переменных при **пошаговой** отладке программы используются окна «Watch» (рис. 1).



Expression	Value	Location	Type
test	1	R4	int
data	2	R5	int
set	10	R6	int
<click to add>			

Рис. 1 – Окно просмотра значений переменных.

Чтобы постоянно получать данные из **глобальных** переменных во время **непрерывной** работы программы используется окно «Live Watch» (рис. 2).



Expression	Value	Location	Type
setE5	0	0x20000000	bool
setE6	0	0x20000001	bool
<click to add>			

Рис. 2 – Окно просмотра изменяющихся переменных.

Для изменения скорости получения данных можно задавать интервал обновления в миллисекундах «Tools» -> «Options...» (рис. 3).

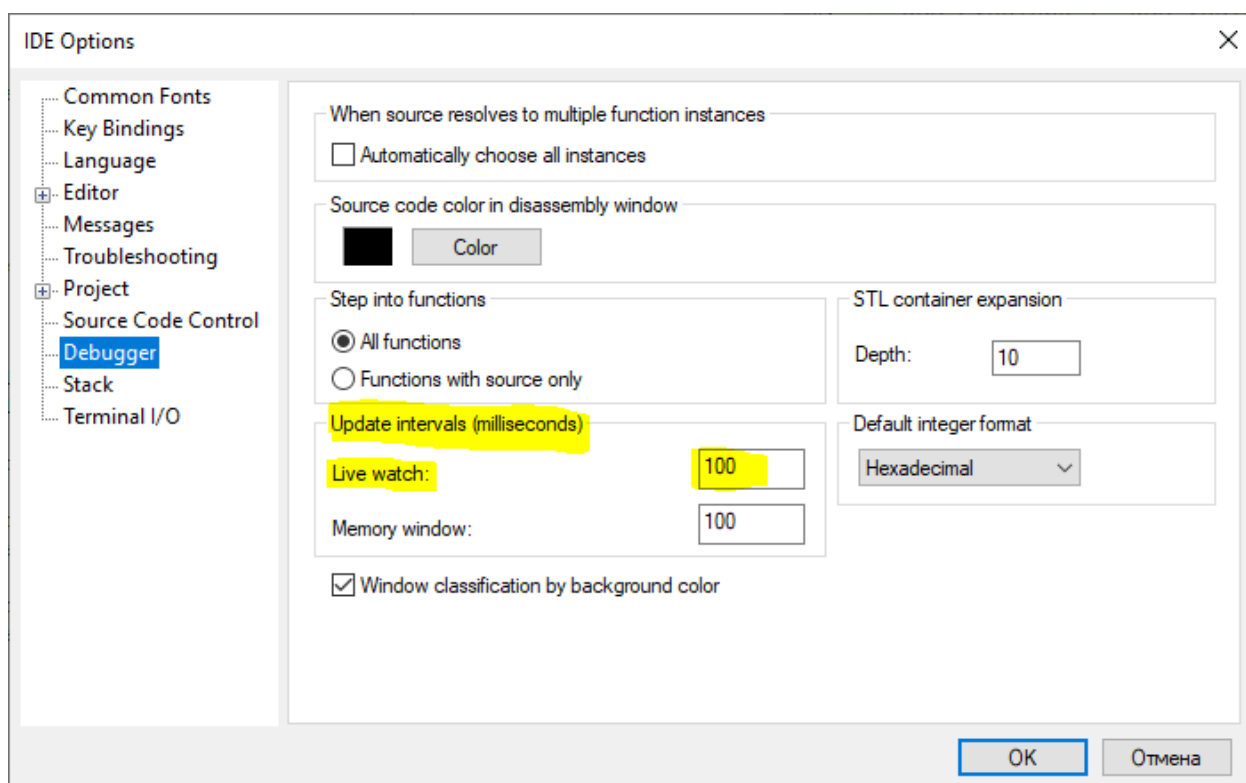


Рис. 3 – Окно настройки скорости обновления.

Добавить переменные в «Watch» и «Live Watch» можно с помощью контекстного меню, вызываемого при выборе нужной переменной (рис. 4).

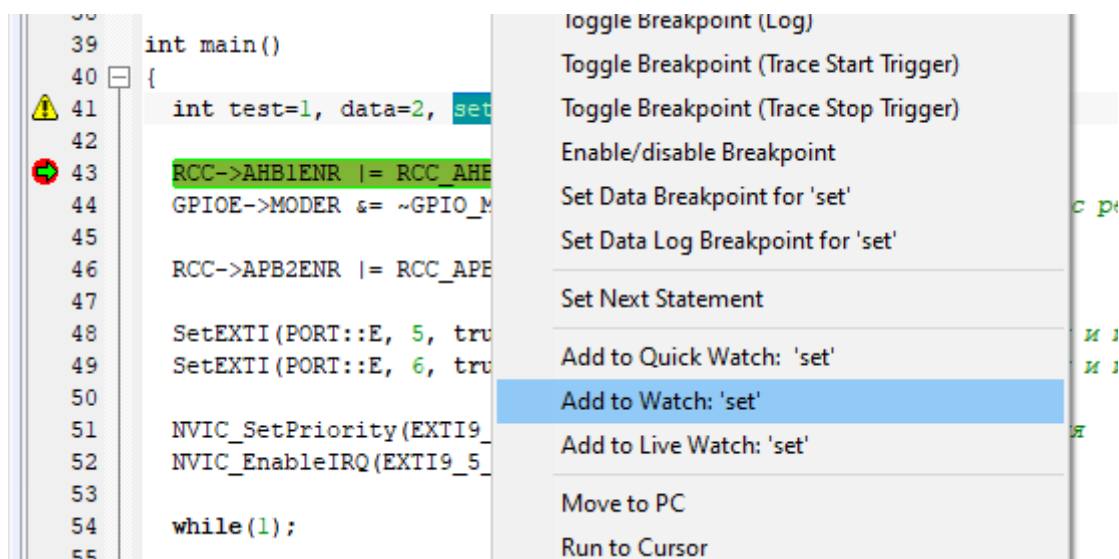


Рис. 4 – Меню добавления в список «Watch».

Чтобы отобразить значения переменных в удобном формате используют контекстное меню настройки «Watch» или «Live Watch» (рис. 5).

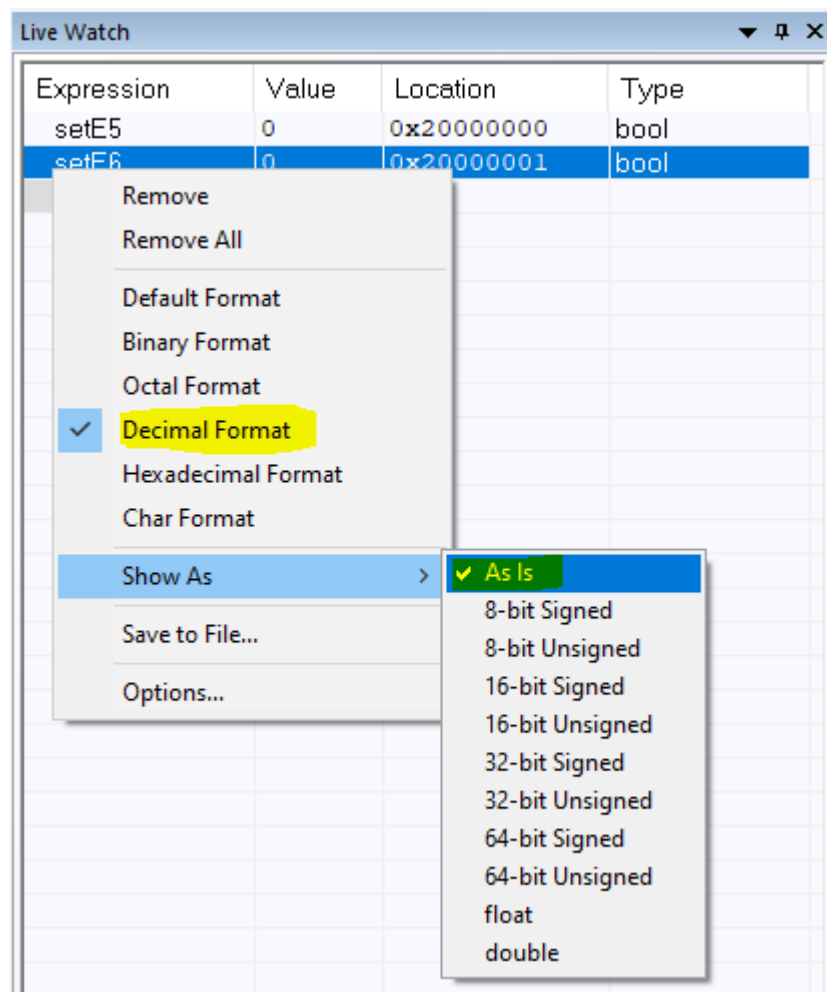


Рис. 5 – Настройка формата и представления данных переменной.

2 Задание к лабораторной работе

- 1) Получить показания потенциометров и джойстика.
- 2) Сгенерировать аналоговый сигнал: пила, треугольник, синусоида и трапеция. Использовать осциллограф для отображения аналогового сигнала.
- 3) Написать функцию управления светодиодом RGB с возможностью задавать цвет и частоту мерцания. Использовать осциллограф для отображения ШИМ сигнала.
- 4) Использовать прерывания от кнопок для переключения светодиодов.