**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Лабораторная работа № 4  
Дисциплина: ЭВМ и периферийные устройства  
Тема: «Изучение принципов обработки прерываний на примере управления встроенными в микроконтроллер таймерами-счетчиками и компаратором»

Выполнил: ст. группы ВТ-31

Проверил: Шамраев А. А.

Белгород 2020

# **Цель работы:** изучить принципы разработки процедур обработки прерываний в микроконтроллере MSP430F1xxx, ознакомиться с принципами функционирования встроенных в микроконтроллер 16 – разрядных таймеров-счетчиков и компаратора для измерения сопротивления резистивного датчика.

# **Задание:** Разработать в среде программирования IAR Embedded Workbench программу на языке С для микроконтроллера MSP430, которая обеспечивает измерение сопротивления переменного резистора и выводит рассчитанное значение на ЖКИ.

# Для решения задачи необходимо использовать встроенный компаратор и таймер А в режиме захвата.

# **Порядок выполнения работы**

– включить лабораторный макет.

– запустить компилятор IAR Embedded Workbench.

– создать пустой проект.

– создать файл ресурса для кода программы и подключить его к проекту.

– ввести код исходного модуля программы измерения сопротивления переменного резистора.

– выполнить компиляцию исходного модуля программы и устранить ошибки, полученные на данном этапе.

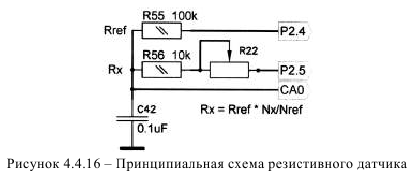
– настроить параметры программатора.

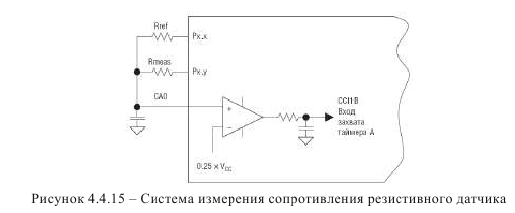
– создать загрузочный модуль программы и выполнить программирование микроконтроллера.

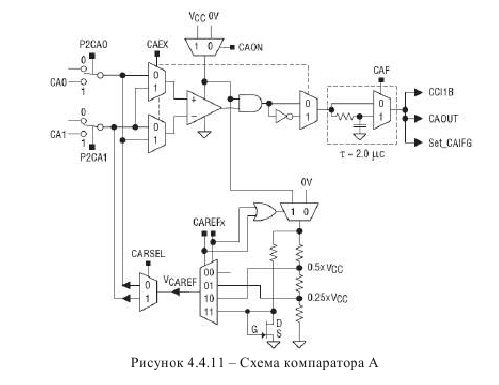
– поверить работоспособность загруженной в микроконтроллер программы и показать результаты работы преподавателю.

В случае некорректной работы разработанной программы, выполнить аппаратный сброс микроконтроллера, провести отладку исходного модуля программы и заново проверить функционирование программы.

**Схема подключения используемых устройств:**







**Текст программы:**

**main.c:**

#include <msp430.h>

#include "stdio.h"

#include "system\_define.h"

#include "system\_variable.h"

#include "function\_prototype.h"

#include "main.h"

/\*

\* main.c

\*/

char message[32] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

void main(void) {

WDTCTL = WDTPW+WDTHOLD;

Init\_System\_Clock();

Init\_System();

\_enable\_interrupt();

LCD\_init();

word R = 0;

while (1) {

R = R22\_get\_resistance();

// Записываем в строку сопротивление

sprintf(message, "%u", R);

LCD\_clear();

// выводим строку

LCD\_message(message);

wait\_1ms(1000);

}

}

**analogsensors.c:**

// Analog sensors functions

#include "function\_prototype.h"

#include "sysfunc.h"

#include "analogsensors.h"

const float HIH\_zero\_offset = 0.958; // параметр "начальное смещение" датчика влажности, В

const float HIH\_slope = 0.03068; // параметр "угол наклона датчика", В / %RH

const float HIH\_ion = 3.3; // опорное напряжение, В

const float HIH\_divisor = 1.1; // коэффициент резистивного делителя

const float INA\_RS = 0.21; // измерительное сопротивление, Ом

const float INA\_RL = 30.1; // сопротивление нагрузки, Ом

// Получить значение относительной влажности, %RH

float HIH\_get\_hum(){

P6SEL |= BIT0; // выбор для ножки P6.0 функции АЦП ADC0, к которому подключен датчик влажности

ADC12CTL1 = SHP + CSTARTADD\_0; // таймер выборки и стартовый адрес преобразования - ADC12MEM0

// выбор опорного напряжения - Vr+ = VеREF+ = 3.3В, Vr- = AVss = 0В

// и входного канала ADC0 для ячейки памяти ADC12MEM0

ADC12MCTL0 = SREF\_3 + INCH\_0;

ADC12CTL0 = ADC12ON; // включение АЦП

ADC12CTL0 |= ENC; // преобразование разрешено

ADC12CTL0 |= ADC12SC; // запуск преобразования

while ((ADC12IFG & BIT0)==0); // ожидание результата преобразования

// пересчет результата преобразования АЦП в значение влажности

// с учетом делителя и опорного напряжения

float rh = (((ADC12MEM0/4095.0) \* HIH\_ion \* HIH\_divisor) - HIH\_zero\_offset) / HIH\_slope;

ADC12CTL0 = 0; // выключение АЦП

return rh;

}

// Получить значение тока потребления системы, А

float INA\_get\_curr(){

P6SEL |= BIT1; // выбор АЦП ADC1, к которому подключен датчик тока

ADC12CTL1 = SHP + CSTARTADD\_1; // таймер выборки и стартовый адрес преобразования - ADC12MEM1

// выбор опорного напряжения - Vr+ = VеREF+ = 3.3В, Vr- = AVss = 0В

// и входного канала ADC1 для ячейки памяти ADC12MEM1

ADC12MCTL1 = SREF\_3 + INCH\_1;

ADC12CTL0 = ADC12ON; // включение АЦП

ADC12CTL0 |= ENC; // преобразование разрешено

ADC12CTL0 |= ADC12SC; // запуск преобразования

while ((ADC12IFG & BIT1)==0); // ожидание результата преобразования АЦП ADC1

// пересчет результата преобразования АЦП в значение тока потребления системы

// с учетом измерительного сопротивления и сопротивления нагрузки:

float curr = (ADC12MEM1\*3.3) / (4095.0 \* INA\_RS \* INA\_RL);

ADC12CTL0 = 0; // выключение АЦП

return curr;

}

// Получить значение сопротивления подстроечного резистора R22, Ом

word R22\_get\_resistance()

{

P2SEL &= ~(Rref+Rx); // функция ввода-вывода для ножек P2.4 и P2.5

word Nref = res\_measure(Rref); // время разряда через опорный резистор

word Nx = res\_measure(Rx); // время разряда через подстроечный резистор

return ((100000\*Nx)/Nref)-10000; // R22 = (100000 \* Nx / Nref) - 10000

}

// Измерение времени разряда конденсатора через resistor (Rref или Rx)

word res\_measure(byte Rpin)

{

P2DIR &= ~Rx; // отключить Rx от конденсатора (направление - ввод)

// заряд конденсатора через опорный резистор Rref

CAPD = ~Rref; // отключение аналоговых сигналов от порта компаратора

P2DIR |= Rref; // подключить Rref к конденсатору (направление - на вывод)

P2OUT |= Rref; // установка ножки Rref- заряд кондесатора

TACCR1 = 65000; // время заряда

TACCTL1 = CCIE; // разрешить прерывания

// тактирование от SMCLK, делитель /4, очистка счетчика, непрерывный режим счета

TACTL = TASSEL\_2 + ID\_2 + TACLR + MC\_2;

LPM0; // перейти в режим пониженного потребления и ожидать прерывания

CACTL2 = P2CA0 | CAF; // вход компаратора подключается к CA0, вкл.выходного фильтра

// включение компаратора, опорное напр. 0.25\*Vcc прикладывается к "-"

CACTL1 = CARSEL+CAREF\_1+CAON;

CAPD = ~(Rpin+CA0);

P2DIR &= ~Rref; // отключить Rref от конденсатора (направление - ввод)

P2DIR |= Rpin; // будем разряжать через ножку Rpin

P2OUT &= ~Rpin; // низкий уровень на Rpin - разряд конденсатора

// захват по заднему фронту, входной сигнал - CCI1B, режим захвата, прерывания разрешены

TACCTL1 = CM\_2+CCIS\_1+CAP+CCIE;

TACTL |= TACLR; // сбросить счетчик таймера

LPM0; // перейти в режим пониженного потребления и ожидать прерывания

TACTL = 0x00; // остановить таймер

CACTL1 = 0x00; // отключить компаратор

CAPD = 0; // включить входные буферы компаратора

return TACCR1; // возвращаем значение счетчика таймера

}

// обработчик прерываний от таймера

#pragma vector=TIMERA1\_VECTOR

\_\_interrupt void isrTIMERA(void)

{

LPM0\_EXIT; // выход из LPM0

TACCTL1 &= ~CCIFG; // очистка флага прерывания

}

**Полученные результаты:** Разработанная программа обеспечивает измерение сопротивления переменного резистора и выводит рассчитанное значение на ЖКИ.

**Выводы:** изучил принципы разработки процедур обработки прерываний в микроконтроллере MSP430F1xxx, ознакомился с принципами функционирования встроенных в микроконтроллер 16 – разрядных таймеров-счетчиков и компаратора для измерения сопротивления резистивного датчика.