Санкт- Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет по курсовой работе
Курс: «Автоматизация проектирования дискретных устройств»
Тема: «Обработка данных с датчика температуры»

Выполнил:
Бояркин Н.С. группа 43501/3
Проверил:
Федотов А.А.

Санкт – Петербург
2016

Оглавление

1. Цель работы

1) Ознакомление со строением систем на кристалле на примере платы MAX 10 NEEK.
2) Ознакомление с прошивкой и методами программирования систем на кристалле.

2. Техническое задание

1) На экране платы должно отображаться численное значение температуры и влажности воздуха, взятое с датчика.
2) Температура должна правильно определяться в пределах от -30 до 50 градусов по Цельсию. Влажность воздуха должна выводиться в процентах.
3) Температура должна выводиться в градусах Цельсия, Фаренгейта и Кельвина. Переключение между ними должно осуществляться через равные промежутки времени или по нажатию на кнопку.
4) Температура и влажность воздуха должны быть выведены с точностью до трех знаков после запятой.
5) На экране платы должны выводиться две динамически меняющихся шкалы для температуры и влажности воздуха соответственно. Наполненность и цвет этих шкал должны меняться с тем же интервалом, что и данные с датчиков.

3. Обзор отладочной платформы

MAX 10 NEEK - полнофункциональная отладочный набор для встраиваемых решений на основе ПЛИС MAX 10 FPGA +NIOS II. MAX 10 NEEK поставляется как интегрированная платформа, включающая аппаратные средства, средства разработки, интеллектуальную собственность и референс-дизайн для разработки широкого спектра аудио, видео и многих других приложений. Полностью интегрированный набор, «всё в одном», MAX 10 NEEK сочетает в себе LCD с 5-точечной емкостной сенсорной панелью и цифровой модуль, обеспечивающий разработчику идеальную платформу для мультимедиа приложений, оптимально используя FPGA.

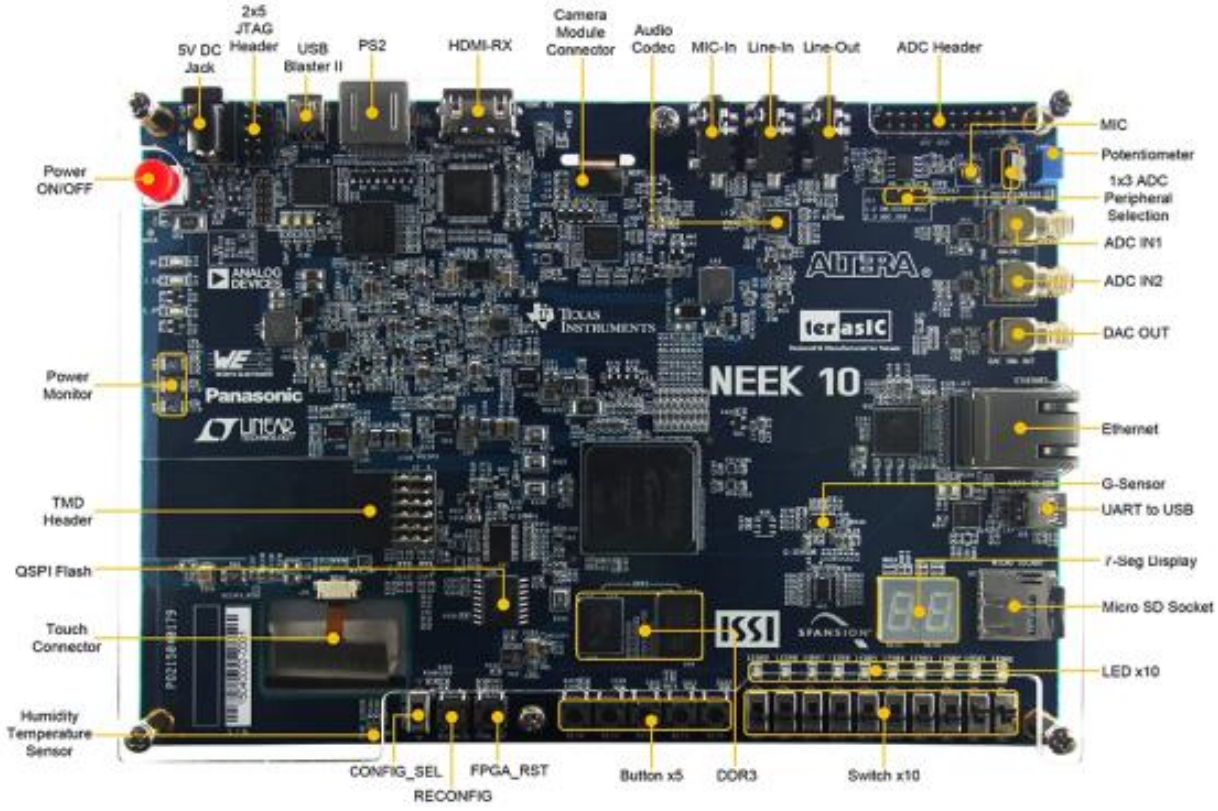

Рис. 1. Плата MAX 10 NEEK

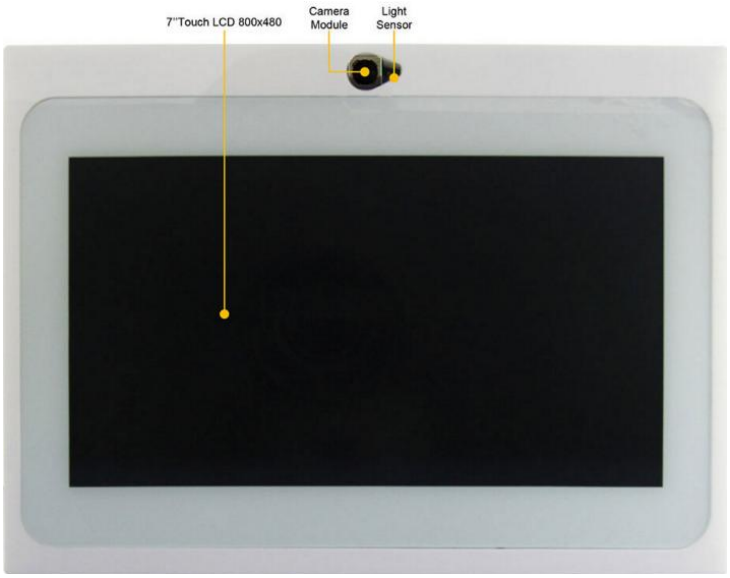

Рис. 2. Передняя панель

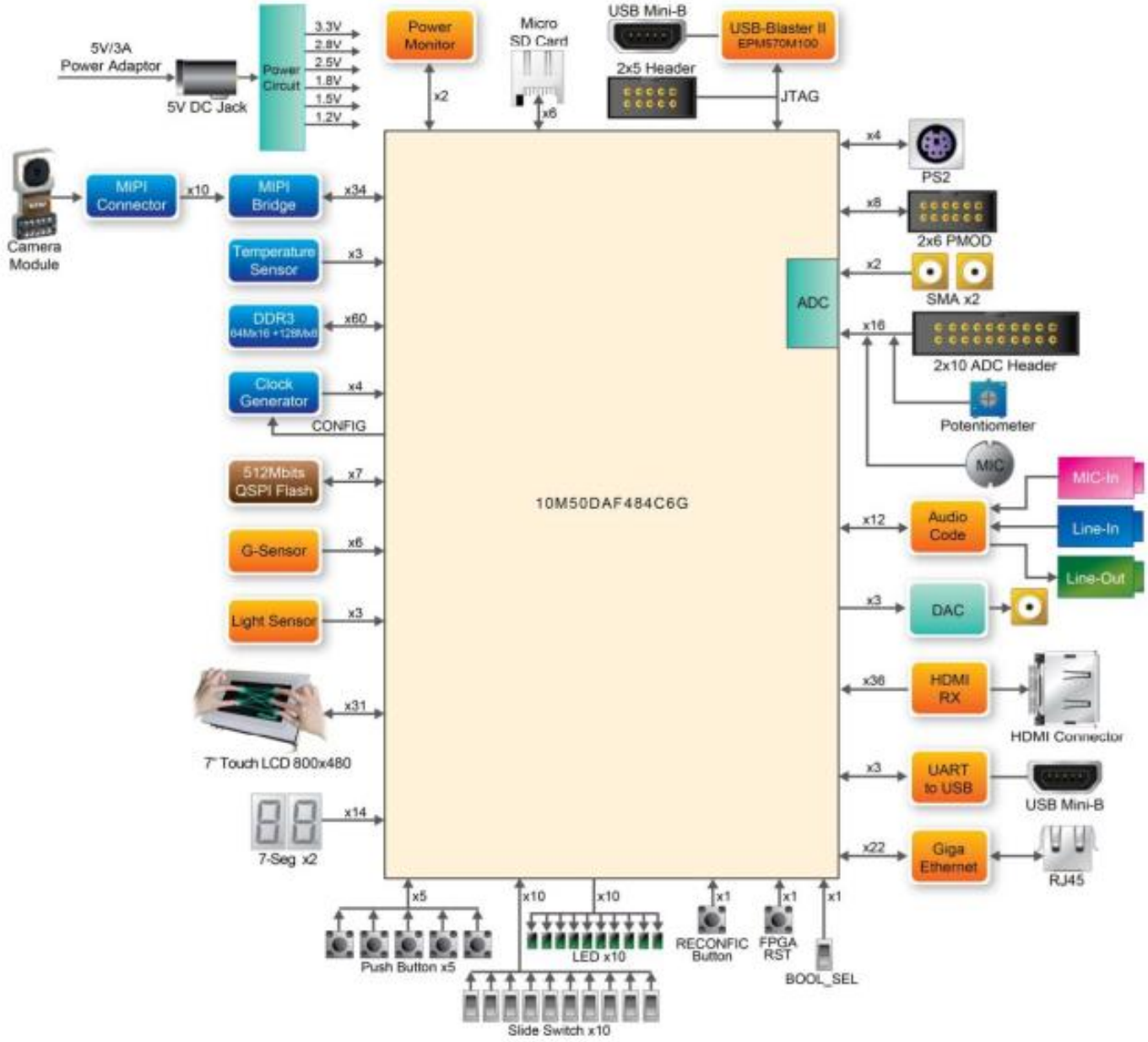

Рис. 3. Подключения устройств

ПЛИС FPGA:

- MAX 10 10M50DAF484C6G;
- интегрированный сдвоенный ADC;
- 50K программируемых логических элементов;
- 1,638 Kbit M9K памяти;
- 5,888 Kbits пользовательской Flash памяти;
- 144 18 x 18 Multiplier;
- 4 PLLs.

Программирование и конфигурирование:

- на плате установлен USB Blaster II;
- опционально программирование непосредственно через 10-контактный разъем JTAG;
- переключатель загрузки.

Память:

- 64Mx16 1 Gb DDR3 SDRAM;
- 128Mx8 1 Gb DDR3 SDRAM;
- 512Mb QSPI Flash;
- слот для Micro SD card.

Коммуникационные интерфейсы и разъем расширения:

- Gigabit Ethernet PHY с разъемом RJ45;
- конвертер UART - USB;
- разъем PS/2: мышь/ клавиатура;
- разъем расширения 2x6 TMD (Terasic Mini Digital).

Дисплей:

- диагональ 7’’, 800x480 цветной LCD с 5-точечной емкостной сенсорной панелью.

Аудио:

- 24-bit CD-качества аудио CODEC с микрофонным входом, линейными входом и выходом.

Видеовход:

- разъем интерфейса HDMI;
- разъем для 8 Мегапиксельной MIPI CSI-2 цветной камеры.

Аналоговые интерфейсы:

- два MAX 10 FPGA ADC SMA входа;
- потенциометр на входе ADC;
- микрофон, установленный на плате, подключен к ADC;
- разъем 2x10 с подключенными входами АЦП MAX 10 FPGA;
- выход ЦАП.

Датчики:

- влажности и температуры;
- датчик освещенности;
- акселерометр;
- монитор питания.

Тактирование:

- один 10 MHz внешний генератор;
- три 50 MHz внешних генератора;

Переключатели, кнопки, светодиоды:

- пять кнопок;
- десять ползунковых переключателей;
- десять красных пользовательских светодиодов;
- два 7-сегментных индикатора.

Питание:

- 5 V/3 A DC.

С учетом всех вышеперечисленных особенностей, данное устройство может быть использовано для реализации встраиваемых приложений, мультимедиа, систем контроля доступа, систем мониторинга и управления.

4. Обзор базового проекта

В основу обработчика данных температурного сенсора лег демонстрационный проект humidity_temperature_lcd. Базовый проект включает в себя: вывод температуры в градусах Цельсия в целочисленном формате, заголовок и несколько картинок.

Рис. 4. Демонстрационный проект humidity_temperature_lcd

Реализация проекта состоит из аппаратной и программной части. Аппаратная часть состоит из файлов Verilog и Qip. эти файлы генерируются инструментом системной интеграции Qsys. Проект в Qsys представлен в виде модулей со входами, выходами и их соединениями между собой.

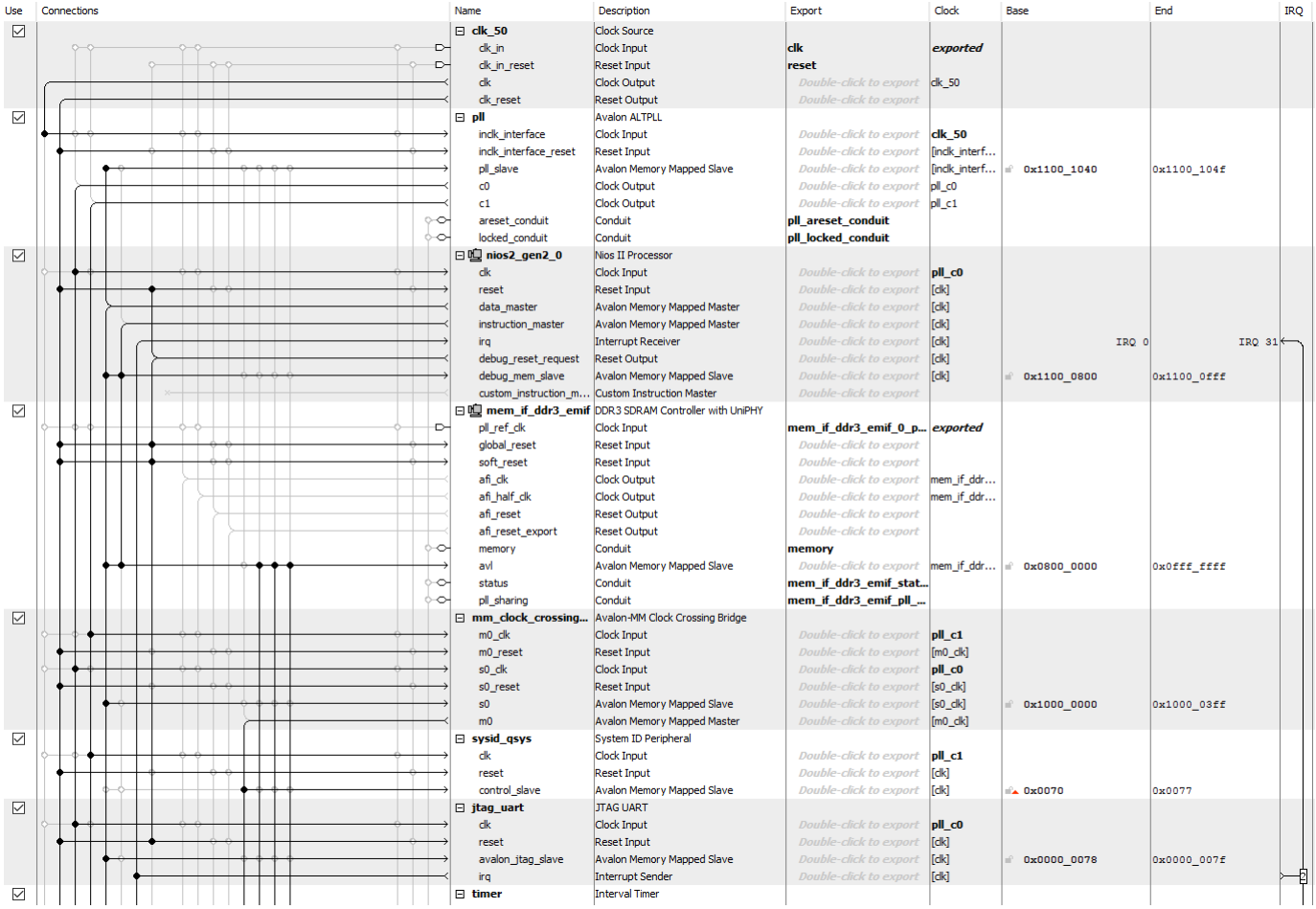

Рис. 5. Подключаемые модули и их соединение Qsys

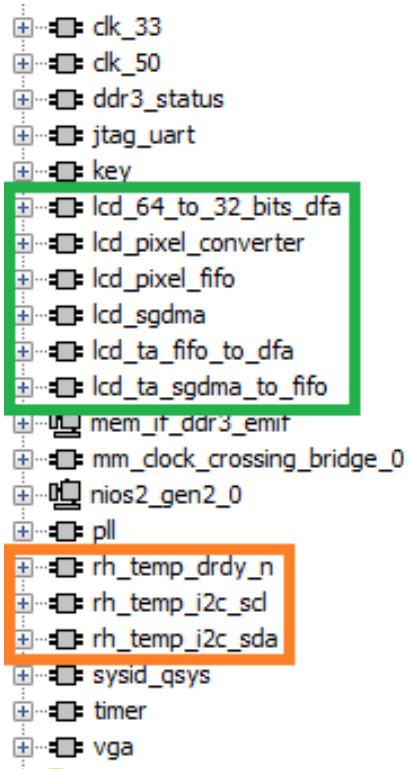

Рис. 6. Подключаемые модули и их соединение Qsys

Основополагающие интерфейсные модули (Рис. 6), которые обеспечивают функциональность проекта выделены зеленым цветом (экран) и оранжевым цветом (датчик температуры и влажности воздуха). Важно отметить, что отсутствуют модули обработки нажатий, поэтому для создания переключения единицы измерения температуры (описанного в техническом задании), с помощью кнопок, потребуются дополнительные подключения.

Рассмотрим функции остальных модулей проекта humidity_temperature_lcd:

- **clk_33**, **clk_50** – генераторы импульсов 33 МГц и 50 МГц соответственно.
- **mem_id_ddr3_emif**, **ddr3_status** – контроллер памяти DDR3 SDRAM и модуль статуса памяти.
- **jtag_uart** – ядро JTAG UART (компонент, предназначенный для передачи данных из Nios в IDE для отладки).
- **key** – кнопки.
- **lcd…** - модули, обеспечивающие вывод на экран.
- **mm_clock_crossing_bridge** – Avalon-MM Clock Crossing Bridge (для обеспечения асинхронной работы счетчиков).
- **nios2_gen2** – программный процессор NIOS II.
- **sysid_qsys** – порт-константа, которой при каждой сборке SOPC присваивается значение, указанное пользователем, и временная метка. Компонент предназначен для защиты от программирования неправильно сконфигурированной системы.
- **timer** – таймер.

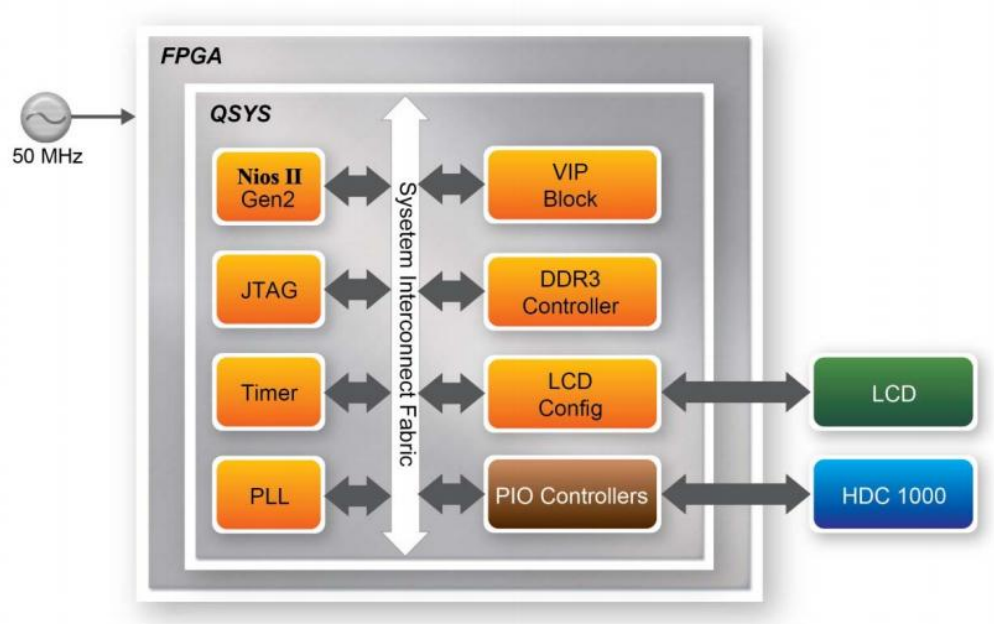

Рис. 7: Диаграмма системных блоков проекта humidity_temperature_lcd

5. Разработка программы

Программная часть состоит из двух проектов: humidity_temperture_lcd и humidity_temperture_lcd_bsp. Второй проект является побочным и необходим для генерации bsp файла. Этот файл содержит интегрированный пакет драйверов и модулей операционной системы, реализующий поддержку определенной аппаратной платформы.

Основной проект humidity_temperture_lcd содержит основной код программы. При сборке этого проекта генерируется программный файл с расширением elf, который впоследствии загружается на устройство.

Одним из требований является организация вывода в разных единицах измерения. Для этого был организован модуль, который определяет значения флагов. Этот вариант предусматривает определение единицы измерения, в зависимости от количества итераций. Попытка изменения этих флагов с помощью кнопок приведена далее.

```
// Переменные для определения единицы измерения
int isCelsius = TRUE;
int isFahrenheit = FALSE;
int isKelvin = FALSE;

// Инициализация остальных переменных
...

// Счетчик количества итераций
int ind = 0;

// Основной цикл
while(1) {

        // Определение единицы измерения, в зависимости от количества итераций
        isCelsius = !(ind % 3);
        isFahrenheit = !((ind + 1) % 3);
        isKelvin = !((ind + 2) % 3);

        ++ind;

        // Получение и вывод информации с датчика
        ...
}
```

Листинг 1. Определение единицы измерения, в зависимости от количества итераций

Далее происходит получение данных с датчика функцией sensor_getdata. Эта функция записывает данные в глобальные переменные temp и rh. Значение температуры temp лежит в границах от -30 до 50 градусов по Цельсию, как и указано в техническом задании. Значение влажности воздуха rh изменяется в пределах от 0 до 100 и измеряется в процентах, что также соответствует техническому заданию.

```
// Основной цикл
while(1) {
        // Определение единицы измерения
        ...

        // Получение информации с датчика в переменные temp и rh
        if(!sensor_getdata()){
            printf("sensor_getdata failed!\n");
            return 0;
        }

        // Обработка и вывод данных с датчика
        ...
}
```

Листинг 2. Получение информации с датчика

После этого, определяется значение температуры, в зависимости от ранее установленных флагов и выводится на экран. Значение температуры выводится с точностью до трех знаков после запятой, что соответствует техническому заданию.

```
// Основной цикл
while(1) {
        // Определение единицы измерения
        ...

        // Получение информации с датчика
        ...

        // В зависимости от единицы измерения, определяем выводимую температуру
```

```
        if(isCelsius) {
            resultTemperature = temp;
            strcpy(postfix, " C");
        }
        else if(isFahrenheit) {
            resultTemperature = temp * 1.8 + 32;
            strcpy(postfix, " F");
        }
        else if(isKelvin) {
            resultTemperature = temp + 273.15;
            strcpy(postfix, " K");
        }

        // Выводим температуру
        snprintf(szText, 128, "%4.3f", resultTemperature);
        vid_print_string_alpha(30, 30, COLOR_Temp_Up, COLOR_Temp_Down, tahomabold_32, display, szText);
        int length = vid_string_pixel_length_alpha(tahomabold_32, szText);

vid_print_string_alpha(30 + length, 30, COLOR_Temp_Up,COLOR_Temp_Down, tahomabold_32, display, postfix)
;

        // Вывод шкал, данных влажности воздуха
        ...
}
```

Листинг 3. Вывод температуры на экран

Определение заполенности шкалы температуры определяется простой пропорцией: нам известны максимальные и минимальные значения температуры, также известны минимальные и максимальные значения на шкале.

В зависимости от температуры, шкала окрашивается в определенный цвет. На промежутке [-30, -10) – синий, [-10, 10) – голубой, [10, 27) – желтый, [27, 37) – оранжевый, [37, 50] – красный.

```
// Основной цикл
while(1) {
        // Определение единицы измерения
        ...

        // Получение информации с датчика
        ...

        // Вывод значения температуры
        ...

        // Определение заполненности шкалы температуры
        int coordinate;
        if(temp < -30)
            coordinate = 32;
        else if(temp > 50)
            coordinate = 152;
        else
            coordinate = (int) ((temp + 30.) / 80. * 120. + 32.);

        // Определение цвета шкалы температуры
        int color;
        if(temp < -10)
            color = BLUE_24;
        else if (temp < 10)
            color = AQUA_24;
        else if (temp < 27)
            color = YELLOW_24;
        else if (temp < 37)
            color = ORANGE_24;
        else if (temp >= 37)
            color = RED_24;
```

```
        // Вывод шкалы температуры
        vid_draw_box(30, 89, 153, 122, BLACK_24, DO_NOT_FILL, display);
        vid_draw_box(31, 90, coordinate, 121, color, DO_FILL, display);


        // Вывод влажности воздуха и шкалы влажности воздуха
        ...
}
```

Листинг 4. Прорисовка шкалы температуры

Вывод численного значения влажности воздуха производится аналогично выводу температуры. Точность в три знака после запятой соответствует техническому заданию.

```
// Основной цикл
while(1) {
        // Определение единицы измерения
        ...

        // Получение информации с датчика
        ...

        // Вывод численного значения и шкалы температуры
        ...

        // Вывод значения влажности воздуха
        snprintf(szText, 128, "%4.3f", rh);
        vid_print_string_alpha(30, 130, COLOR_Rh_Up, COLOR_Rh_Down, tahomabold_32, display, szText);
        length = vid_string_pixel_length_alpha(tahomabold_32, szText);

vid_print_string_alpha(30 + length, 130, COLOR_Rh_Up, COLOR_Rh_Down, tahomabold_32, display, "
%RH");

        // Определение заполненности шкалы влажности воздуха
        if(rh < 0)
            coordinate = 32;
        else if(rh > 100)
            coordinate = 152;
        else
            coordinate = (int) (rh / 100. * 120. + 32.);

        // Определение цвета шкалы влажности воздуха
        if(rh < 30)
            color = YELLOW_24;
        else if (rh < 70)
            color = AQUA_24;
        else if (rh >= 70)
            color = BLUE_24;
}
```

Листинг 5. Вывод влажности воздуха и прорисовка шкалы влажности

Дополнительно была попытка реализовать выбор единицы измерения с помощью кнопок. Код обработки касаний был полностью перенесен из демонстрационного проекта lcd_painter.

```
// Инициализация касаний
TC2_INFO *pTouch = MTC2_Init(I2C_OPENCORES_0_BASE, LCD_TOUCH_INT_BASE, LCD_TOUCH_INT_IRQ);
if (!pTouch){
    printf("Failed to init multi-touch\r\n");
} else{
    printf("Init touch successfully\r\n");
}


// Переменные для определения единицы измерения
int isCelsius = TRUE;
int isFahrenheit = FALSE;
int isKelvin = FALSE;

// Прямоугольники, для определения попадания
```

```
RectSet(&celsiusButton, 400, 700, 100, 160);
RectSet(&fahrenheitButton, 400, 700, 200, 260);
RectSet(&kelvinButton, 400, 700, 300, 360);


// Основной цикл
while(1) {

        // Получаем информацию о количестве касаний

  if(MTC2_GetStatus(pTouch, &Event, &TouchNum, &X1, &Y1, &X2, &Y2, &X3, &Y3, &X4, &Y4, &X5, &Y5)) {
            PtSet(&Pt1, X1, Y1);

            printf("TOUCH\n");

            if(IsPtInRect(&Pt1, &celsiusButton)) {
                // Если было касание первой кнопки, выводим в цельсиях
                isCelsius = TRUE;
                isFahrenheit = FALSE;
                isKelvin = FALSE;
            }
            else if(IsPtInRect(&Pt1, &fahrenheitButton)) {
                // Если было касание второй кнопки, выводим в фаренгейтах
                isCelsius = FALSE;
                isFahrenheit = TRUE;
                isKelvin = FALSE;
            }
            else if(IsPtInRect(&Pt1, &kelvinButton)) {
                // Если было касание третей кнопки, выводим в кельвинах
                isCelsius = FALSE;
                isFahrenheit = FALSE;
                isKelvin = TRUE;
            }
        }

        // Получение и вывод информации с датчика
        ...
}
```

Листинг 6. Попытка обработки нажатия кнопок

Однако при правильном программном коде и успешной сборке проекта, обработка касаний все равно не работает. Это доказывает, что отсутствие подключенного модуля обработки касаний в Qsys действительно существенным образом влияет на функционирование проекта.

6. Аппаратная часть

Qsys файл проекта humidity_temperature_lcd по умолчанию включает в себя набор подключаемых модулей, которые иногда бывают избыточны. Например, для этого проекта незадействованным остается модуль обработки нажатий на кнопки key. Отключим этот модуль и сгенерируем qip файл.

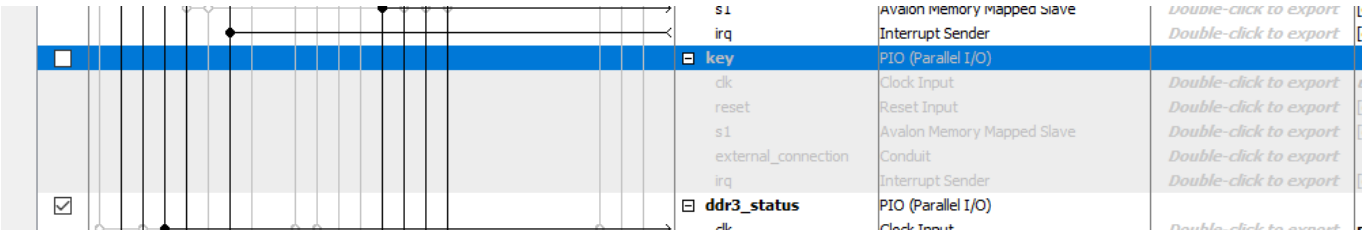

Рис. 7. Отключение модуля key

После этого закомментируем в верилоговском файле humidity_temperature_lcd.v. все упоминания о модуле key. После компиляции проекта был получен обновленный аппаратный файл с расширением sof.

7. Результаты работы программы

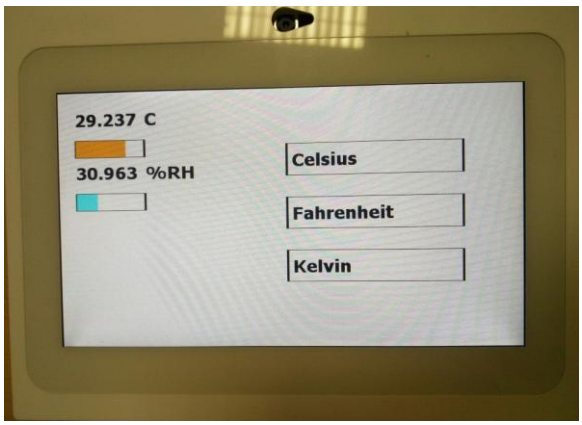

Рис. 8. Результат в Цельсиях

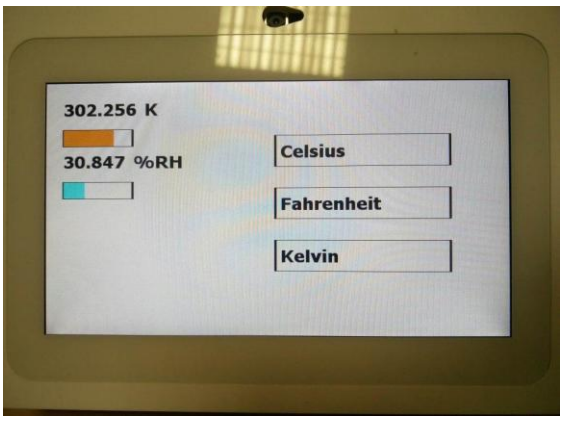

Рис. 9. Результат в Кельвинах

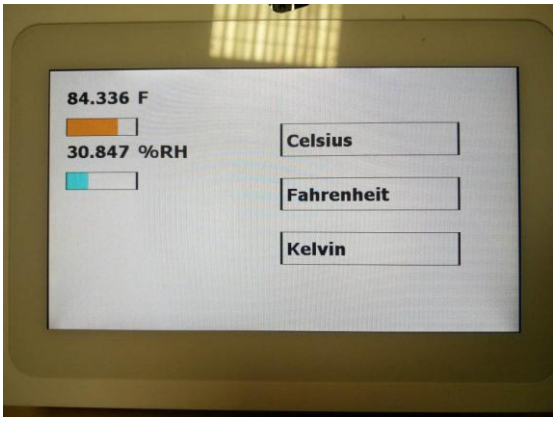

Рис. 10. Результат в Фаренгейтах

8. Вывод

В ходе данной работы, были получены навыки работы и программирования на отладочном комплексе MAX 10 NEEK. В частности, были освоены принципы работы с графической функциями для вывода на экран и функциями для получения данных с датчика температуры.

Также, в полном соответствии с техническим заданием был реализован собственный проект. В ходе его разработки было доказано, что наличие просто программного кода не гарантирует правильную работу проекта. Очень важно подключить те модули, которые необходимы в проекте и отключить те модули, которые не используются.

Наличие множества демонстрационных проектов от разработчиков платы существенно ускоряет разработку. Однако, код этих проектов не стоит использовать для больших проектов, потому что в нем отсутствуют всякие представления о правильном оформлении и структуризации кода.