ДЕПАРТАМЕНТ ОБЗРАНИЯ,НАУКИ И МОЛОДЁЖНОЙ ПОЛИТИКИ
ВОРОНЕЖСКОЙ ОБЛАСТИ
ГБПОУ ВО
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ПРОМЫШЛЕННО
ГУМАНИТРАНЫЙ КОЛЛЕДЖ

Разработка ультразвукового сенсора наполнения бака с помщью STM8S

Постановка задачи

1. Управление погружным вибрационным насосом.
2. Измерение уровня воды в 220 литровой бочке.
3. Включение электромагнитного клапана по запросу — начало полива. Отключение производится по сигналу от измерителя уровня воды.
4. По завершению цикла полива запуск цикла заполнения бочки. Отключение производится по сигналу от измерителя уровня воды.

Состав оборудования

— US-100 — ультразвуковой измеритель расстояния. Старший брат знаменитого HC-SR04. Главное отличие — наличие температурной компенсации и возможность работы в режиме передачи данных по UART. По точности сравнить мне их не удалось ввиду отсутствия HC-SR04.
— Плата, оборудованная микропроцессором STM8S003F3P6.
— LCD 2x16, совместимый с HD44780.
— HLK-PM01 — блочный малогабаритный источник питания типа AC-DC. Входное напряжение 220В переменного тока, выходное 5В 600 мА постоянного тока.
— Электромагнитный клапан с установочным диаметром 3\4 на напряжение 24В постоянного тока. Потребляемый ток достигает 2 А.
— Корпус для основного прибора.
— Корпус для ультразвукового сенсора. Исполнение этого корпуса IP67, и как показала практика, такое исполнение было выбрано не напрасно.

Средства разработки

Была выбрана плата на платформе STM8 с гораздо меньшим количеством ресурсов. В качестве среды программирования и отладки был выбран IAR Embedded Workbench for STM8. Данная среда прекрасно работает с программатором — отладчиком ST-LINK V2. Программатор имеет интерфейс USB и подключается к отлаживаемому изделию всего 4-мя проводами. При этом зачастую тока от отладчика достаточно для питания отлаживаемой платы.

Операционная система

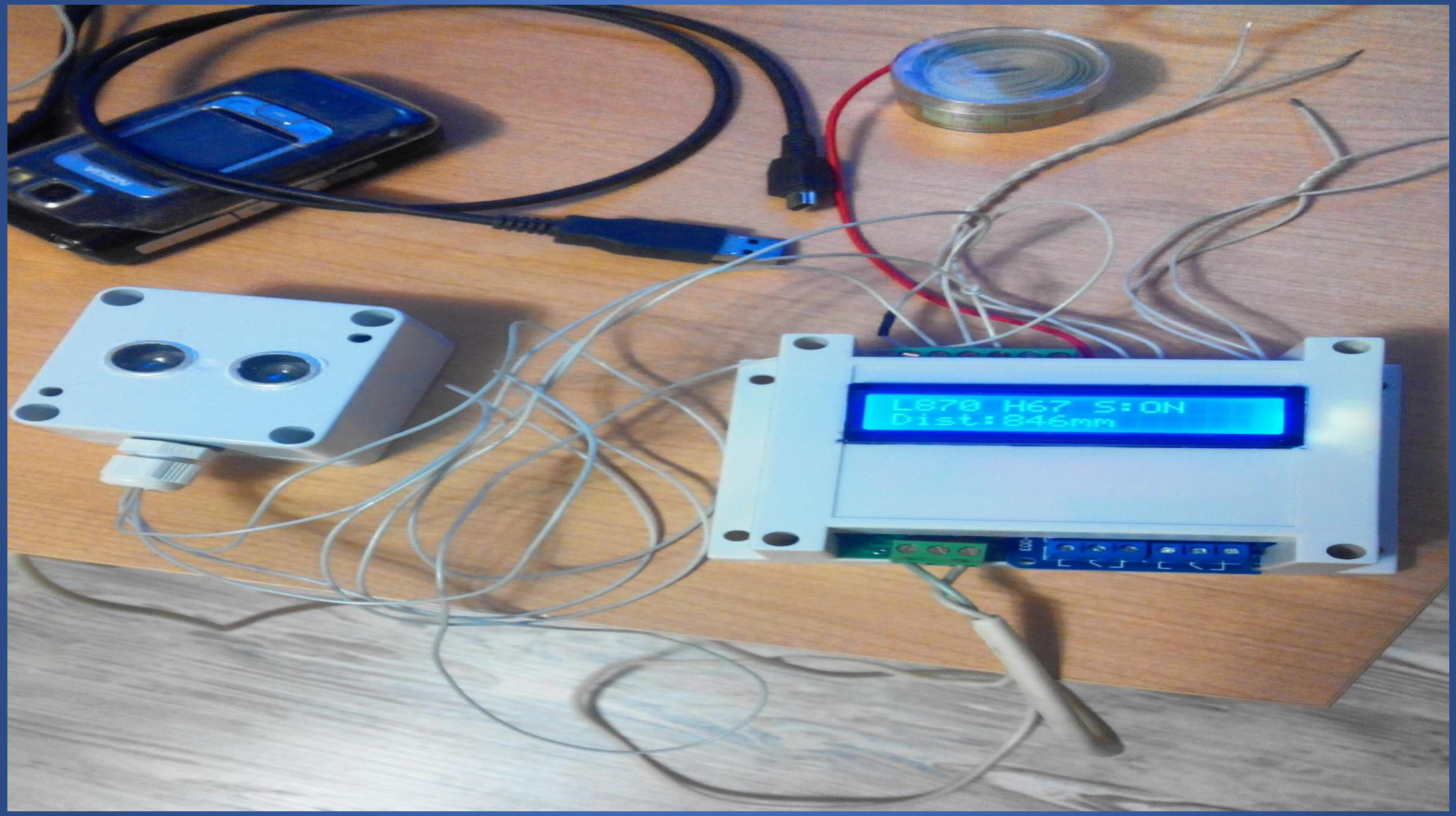
В самом начале пути я встал перед выбором — использовать или нет операционную систему реального времени для микропроцессора, обладающими такими ресурсами. И выбор был сделан весьма неожиданный — ChibiOS RT v2.6.9. Не буду в этой публикации рассматривать все особенности этой системы — только отмечу, что создание двух потоков с одинаковыми приоритетами заняло 2547 байт флеш — памяти и 461 байт оперативной. Собственно, немало — но результатом этой потери стало то, что теперь я имею 8 — разрядный недорогой микроконтроллер, который управляется операционной системой реального времени. И следовательно, я могу управлять исполнением моих задач так, как мне необходимо.

Программирование и сборка

При программировании единственной проблемой было то, что для дисплея и ультразвукового сенсора не было найдено готовых драйверов. Итог — пришлось писать самому. Результатом работы стало стабильно работающая программа.

Внешний вид прототипа, установленного в корпус, показан ниже. Как раз виден процесс тестов в домашних(читайте — тест для сферического процессора в вакууме) условиях. Именно в таких режимах обычно тестируют ардуиноводы, и результатом являются отзывы об исключительной надёжности получаемых «решений». Поведение моего изделия в таком тесте было просто идеальным — никаких сбоев или отклонений замечено не было.

Внешний вид прототипа.



Монтаж и запуск системы

Для монтажа было выбрано строение, расположенное в непосредственной близости от объекта управления. Как я говорил выше, никакой защиты от атмосферных осадков не было предусмотрено. В конце концов, затяжной ливень сделал своё дело — но об этом немного позже. Ниже на картинке показана установка ультразвукового датчика.

Устройство управляет включением системы капельного полива, причём управляется от программируемого логического контроллера. Контроллер имеет встроенную шину 1-wire с возможностью подключения до 128 устройств на один коммуникационный порт. Датчик влажности комбинированный, емкостный, работает на частоте 80 МГц и имеет как раз интерфейс 1-wire. Вместе с влажностью передаёт величину освещённости на уровне установки сенсора.

Эксплуатация

Подав питание, было приятно увидеть, что алгоритм, заложенный в программе, работает так, как и хотел разработчик. Блок измерил расстояние до воды, определил, что бочка пустая, и включил насос для заполнения. В процессе заполнения отклонения показаний датчика уровня составили не более 15 мм, что достаточно приемлемо. Заполнив бочку, отключил насос. Теперь система готова к началу процесса полива.

Характеристики STM8S003F3P6

Ядро 16 МГц, 3-уровневый конвейер
8 Кб flash памяти (20 лет после 100 циклов записи/стирания)
1 Кб SRAM памяти
128 байт EEPROM (100 000 циклов записи/стирания)
10-бит АЦП, 5 каналов
Последовательные интерфейсы – SPI, I2C, UART (IrDA, LIN, Smartcard)
Два 16-бит таймера (ШИМ, захват/сравнение)
Один 8-бит таймер
Питание 2.95 В – 5.5 В
Корпус TSSOP20
От -40 до +85 °C

Инструментарий для работы

Для работы с любым микроконтроллером, и STM8 – в частности, нам понадобятся отладочный комплект и среда разработки с Си-инструментарием. В качестве отладочного комплекта можно использовать любой из предоставляемых комплектов как от ST, так и от сторонних производителей

Выбор программного инструментария для разработки

На данный момент для разработки и отладки программного обеспечения для STM8 существует четыре среды: ST Toolset от STMicroelectronics, Ride7 от Raisonance, CXSTM8 от Cosmic software, IAR Embedded Workbench от IAR Systems. Сравнительный анализ средств разработки программного обеспечения представлен в таблице 1.

Инструментарий	**Среда разработки**	**Си-инструментарий**	**Си-инструментарий других производителей**	**Ограничение Си-иструментария, Кбайт**	**Программатор-отладчик**
STMicroelectronics	ST Visual Develop	Нет	Cosmic software, Raisonance	Нет	ST-Link R-Link STICE
Raisonance	Ride 7	Есть	Нет	16	R-Link
Cosmic software	CXSTM8	Есть	Нет	32	Нет
IAR Systems	IAR Workbench	Есть	Нет	8 или полная версия на 30 дней	ST-Link STIC

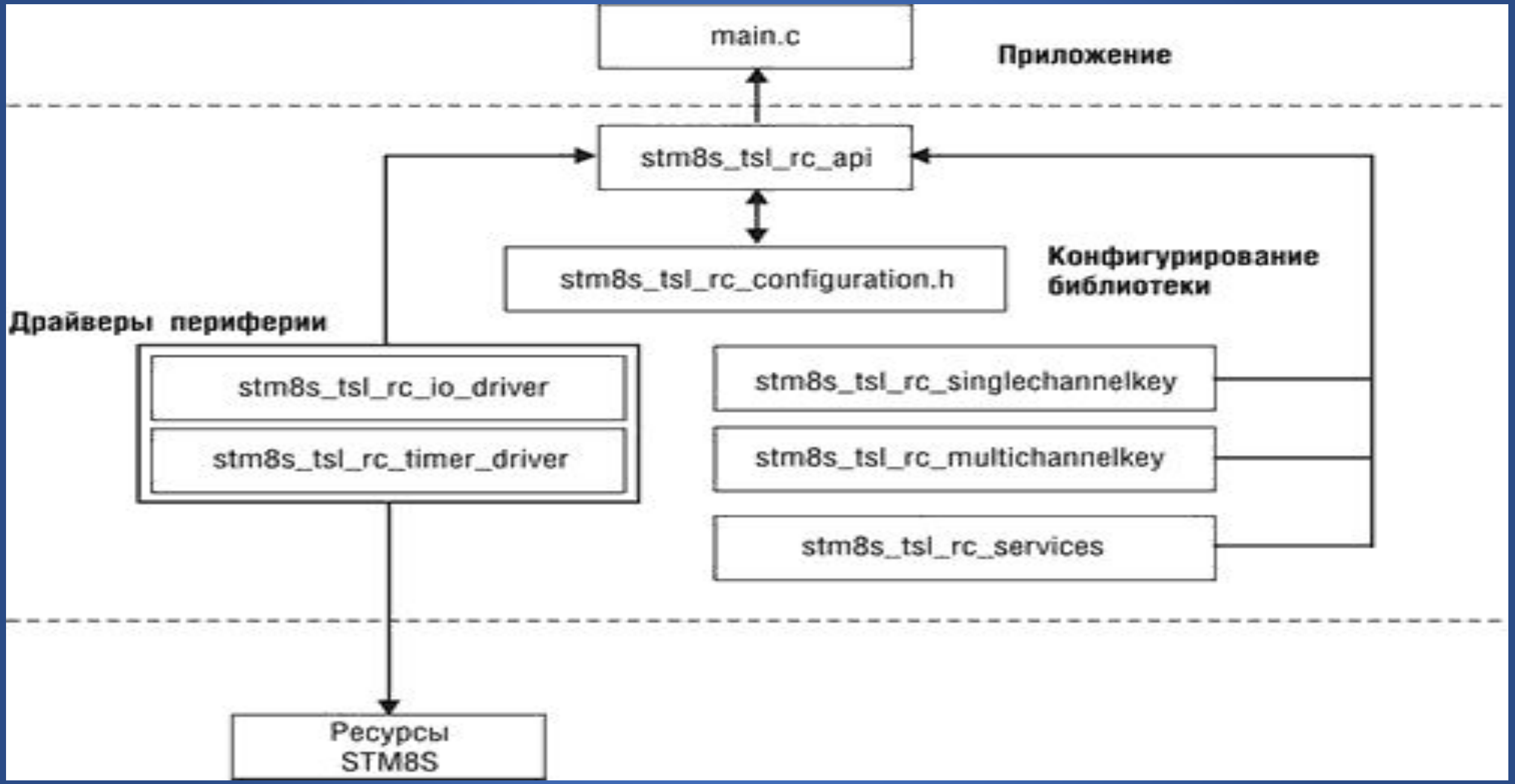
Пакет ST Toolset включает в себя среду разработки ST Visual Develop и отдельную программу для более функционального внутрисхемного программирования flash-памяти микроконтроллеров ST Visual Programmer. Среда разработки ST Visual Develop имеет встроенный инструментарий для разработки программного обеспечения на языке assembler, но у нее также имеется возможность подключения и использования Си-инструментария от Raisonance и Cosmic software. Стоит заметить, что все четыре среды с Си-инструментарием предоставляют возможность использования с некоторыми ограничениями, а именно – по загружаемому коду во flash-память. У Raisonance оно составляет 16 Кбайт, у Cosmic software - 32 Кбайт, а у IAR – 8 Кбайт или полную версию с 30-дневным ограничением. Самый дешевый и оптимальный вариант – это использование ST Visual Develop в качестве среды разработки и Си-инструментария либо от Raisonance, либо от Cosmic software. Вы, конечно же, можете заметить, что наиболее известным и популярным является инструментарий от IAR System, и тут с вами трудно не согласиться. Он представляет собой более серьезный продукт с лучшей технической поддержкой, но и является самым дорогим. Итак, мы останавливаем свой выбор на ST Visual Developer, плюс Си-инструментарий от Cosmic software и Raisonance. Большой разницы в использовании обоих инструментариев нет, и далее в своих разработках вы можете остановиться на любом из них. Если вы – начинающий разработчик, и у вас недостаточный опыт, рекомендую устанавливать все программное обеспечение по предполагаемым установщиком местам расположения, т.е. по умолчанию. Это необходимо для меньшей путаницы при дальнейшей настройке проектов, расположении файлов, библиотек и синхронизации с информацией приведенных материалов.

Установка программного инструментария для разработки

На сайте STMicroelectronics вы можете найти всю необходимую информацию для работы с STM8S. На основе одного из примеров мы рассмотрим проект, более подробно останавливаясь на ключевых моментах. Также мы остановимся на двух библиотеках – сенсорной и библиотеке стандартной периферии микроконтроллера – написанных инженерами STMicroelectronics для быстрого освоения всех линеек МК и вывода продукции на рынок. Вы можете отказаться от использования библиотек и работать напрямую с именами регистров или написать собственное программное обеспечение для работы с периферией микроконтроллера.

Обзор библиотек

Библиотека стандартной периферии содержит набор функций, структур данных и макросов, охватывающих свойства периферии микроконтроллеров STM8S. Использование библиотеки в значительной степени облегчает процесс разработки собственного программного обеспечения, т.к. устраняется необходимость изучения документации с именами регистров и их функционального назначения

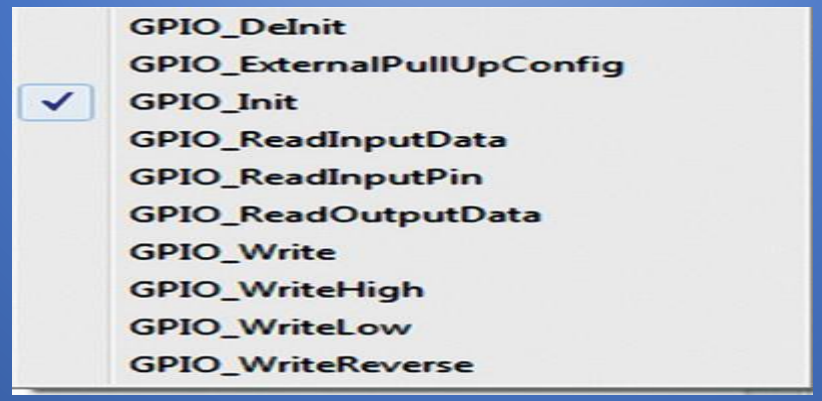


Структура сенсорной Библиотенки

Для использования библиотеки в приложении необходимо добавить все заголовочные и исполняемые файлы в проект. Исключение составляют два файла – «STM8_TSL_RC_Configuration_TOADAPT.h» и «STM8_TSL_RC_routines.asm». Первый копируется в директорию проекта и переименовывается в «stm8_TSL_RC_Configuration.h». Второй добавляется в проект только при использовании инструментария от Raisonance. Файлы «stm8_tsl_rc_api.h» и «stm8_tsl_rc_api.c» определяют функции API, переменные, структуры данных, константы для связи между библиотекой и кодом пользователя. Для использования библиотеки заголовочный файл «stm8_tsl_rc_api.h» должен быть включен в основной исполняемый файл «main.c». Документ «stm8_TSL_RC_Configuration.h» содержит статические конфигурационные параметры, которые должны быть сконфигурированы в соответствии с аппаратной частью проекта. Необходимо проверить все параметры с префиксом «#define» в соответствии с правильными значениями.

Написание своего приложения

Итак, мы вплотную приблизились к написанию собственного кода. Мы напишем программу, которая будет конфигурировать вывод порта, к которому подключен светодиод, и поуправляем им.
Светодиод катодом подключен к PD0, анод подключен через токозадающий резистор к питанию. Таким образом, чтобы включить светодиод, необходимо подать на вывод порта логический 0.
В первую очередь изучите файл-шаблон main.c. Вы найдете в нем, кроме самого основного приложения, интересную функцию void assert_failed(u8* file, u32 line). По умолчанию она отключена макросом USE_FULL_ASSERT. Данная функция позволяет отследить некорректное использование библиотеки в том случае, если вы задали какой-то параметр неверно или перепутали порядок следования параметров. Эту опцию рекомендуется активировать для режима отладки, включив в список преопределенных символов компилятора.
Начинаем модификацию void main(void).
Для того, чтобы понять, какую функцию нужно взять для инициализации из библиотеки, нужно открыть список функций файла stm8s_gpio.c, и станет очевидно, что нужна функция GPIO_Init



Функции драйвера периферийного модуля GPIO

Тут же можно посмотреть, какие аргументы принимает эта функция. Для этого не нужно открыть толстые справочники, а достаточно просто посмотреть описание функции, расположенное над ней самой в виде комментария

```
in.c * | stm8s_conf.h | stm8s_adc2.c | stm8s.h | stm8s_it.c | stm8s_gpio.c

/**
  * @brief  Initializes the GPIOx according to the specified parameters.
  * @param  GPIOx : Select the GPIO peripheral number (x = A to I).
  * @param  GPIO_Pin : This parameter contains the pin number, it can be any value
  *         of the @ref GPIO_Pin_TypeDef enumeration.
  * @param  GPIO_Mode : This parameter can be a value of the
  *         @Ref GPIO_Mode_TypeDef enumeration.
  * @retval None
  */

void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef GPIO_Pin, GPIO_Mode_TypeDef GPIO_Mode)
```

Функция инициализации GPIO

Соответственно, первый аргумент станет очевидно понятными, это GPIOD. А вот для понимания того, что должны из себя представлять остальные, требуется отправиться к определениям GPIO_Pin_TypeDef и GPIO_Mode_TypeDef. И сразу станет ясно, что это GPIO_PIN_0 и GPIO_MODE_OUT_PP_HIGH_FAST соответственно.

Итак, мы получили функцию

GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_HIGH_FAST)

Включим ее в основной код. Вот и вся инициализация. После этого остается только воздействовать на состояние вывода. Для этого также обратимся в файл библиотеки и найдем наиболее подходящую функцию void GPIO_Write(GPIO_TypeDef* GPIOx, uint8_t PortVal). Аналогичным образом найдем, какие должны быть у данной функции аргументы для того, чтобы включить или выключить светодиод.

Итак, в итоге мы получим следующий вид функции void main(void)

```
/* Includes ------------------------------------------------------------------*/
#include "stm8s.h"

/* Private defines -----------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/

void main(void)
{
  GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_HIGH_FAST);
  /* Infinite loop */
  GPIO_Write(GPIOD, 0);
  GPIO_Write(GPIOD, 1);
  GPIO_Write(GPIOD, 0);
  while (1  void GPIO_Write(GPIO_TypeDef *GPIOx, uint8_t PortVal)
  {
  }

}

#ifdef USE_FULL_ASSERT
```

Функция void main(void)

После этого загрузим код в контроллер и по шагам (клавиша F10) пройдем все команды для того, чтобы убедиться, что код работает.

Вывод.

Микроконтроллеры семейства STM8 — это мощные и, в то же время, недорогие устройства, на которых можно строить различную домашнюю и промышленную автоматику. Крайне невысокая цена контроллеров линейки Value Line делает их весьма конкурентоспособными на рынке средств для построения различных датчиков (дыма, газа) там, где массовость, компактный размер и цена важны в равной степени. А то, что контроллер очень удобен в использовании как в электрическом, так и в программном смысле, делает возможным окончить разработку в кратчайшие строки.