

FormFrameWork: импортонезависимая программная платформа для разработки встраиваемых систем управления

Ксения ЛУКИНА,
руководитель проекта, ООО «ФОРМ»
Алексей ТУШЕНЦОВ,
инженер-программист микроконтроллеров,
ООО «ФОРМ»

Статья адресована разработчикам встраиваемых систем, инженерам-электронщикам и всем, кто интересуется ответственными решениями в области проектирования электронной техники.

Введение

Компания ФОРМ впервые представляет свой новый продукт – отечественную программную платформу **FormFrameWork (FFW)**, разработанную для микроконтроллеров и ПЛИС в обеспечение импортонезависимости встроенного программного обеспечения на языке C/C++. Платформа ориентирована на использование в системах с распределенным управлением, например в модульном автоматизированном тестовом оборудовании (АТЕ).

1. Глоссарий терминов в контексте статьи

Встроенное программное обеспечение (ВПО): код на С или С++, выполняющийся на микроконтроллере или ПЛИС из состава объекта разработки и использующий ресурсы платформы FFW.

Платформа FFW: набор системных модулей и библиотек функций, который встраивается в проекты ВПО и представляет собой единый программный интерфейс для ресурсов устройства. Цель платформы FFW – уменьшить объем кода, зависящего от конкретной элементной базы, и облегчить перенос проектов.

Объект разработки: конкретное цифровое устройство или модуль (например, из состава АТЕ по стандарту FormRXI), в составе которого используется ВПО, в отношении которого применяются алгоритмы работы и алгоритмы испытаний.

Прикладной уровень ВПО: часть ВПО, которая реализует функции целевого приложения, исполняющие алгоритмы работы и алгоритмы испытаний объекта разработки на основе прикладного программного интерфейса (Application Programming Interface, API) платформы FFW.

Системный уровень ВПО: условно-постоянная часть ВПО, которая реализуется средствами платформы FFW на стороне устройства и предоставляет прикладному уровню единый программный интерфейс к ресурсам объекта разработки. Включает системные библиотеки функций и базовые сервисы платформы FFW, реализующие типовые алгоритмы работы с ресурсами объекта разработки и предоставляющие прикладному уровню единый программный интерфейс.

Аппаратно-зависимый уровень ВПО: часть ВПО, которая зависит от конкретной элементной базы. Включает в себя адаптеры и низкоуровневые драйверы периферии микроконтроллеров и IP-ядер ПЛИС, использующих SDK и библиотеки производителей и обе-

спечивающих поддержку абстрактных интерфейсов платформы FFW.

Функция: именованный фрагмент программы на С или С++, который можно вызвать и который выполняет определенную операцию, при необходимости возвращая результат.

Библиотека функций платформы FFW: набор модулей платформы FFW, содержащих функции для работы с типовыми задачами. Например: таймеры, очереди, интерфейсы ввода-вывода, энергонезависимая память, обработка команд, логирование, отладка и т. д. Библиотеки функций платформы FFW – это набор системных модулей, функции которых реализуют типовые алгоритмы работы с ресурсами объекта разработки и предоставляют прикладному уровню стандартный API.

Базовые сервисы платформы FFW: системные модули, которые имеют свое состояние и выполняются регулярно в составе суперцикла. Используют библиотеки функций. Обеспечивают непрерывную работу таких типовых механизмов ВПО, как программные таймеры, обмен командами, управление настройками и диагностика.

Системный драйвер: модуль на системном уровне ВПО, функции которого реализуют абстрактный интерфейс платформы FFW для ресурса определенного типа (SPI, таймер, GPIO и т. д.) и используют адаптеры или низкоуровневые драйверы.

Низкоуровневый драйвер: модуль аппаратно-зависимого уровня, который напрямую работает с регистрами периферии конкретного микроконтроллера или IP-ядра ПЛИС и обычно опирается на SDK производителя.

Адаптер: программная «прослойка» между абстрактным интерфейсом платформы FFW и низкоуровневым драйвером. С одной стороны, адаптер реализует интерфейс, ожидаемый библиотеками функций FFW, а с другой, использует SDK конкретного устройства.

Алгоритм: совокупность точно заданных правил решения некоторого класса задач или набор инструкций, описывающих порядок действий исполнителя для решения определенной задачи. В контексте рассматриваемого ВПО алгоритм всегда реализован в виде одной или нескольких функций прикладного или системного уровня.

Алгоритм работы: алгоритм, описывающий штатную функциональность модуля или устройства в составе системы и реализуемый в виде функций прикладного уровня ВПО.

Алгоритм испытаний: алгоритм, описывающий порядок действий для проверки соответствия объекта разработки установленным требованиям. Включает подготовку, воздействие, измерение

и оценку результата. В составе ВПО реализуется в виде функций прикладного уровня.

Управляющая ЭВМ: внешний по отношению к объекту разработки компьютер (ПК, индустриальный компьютер) с установленным специализированным системным программным обеспечением, который управляет объектом разработки и его испытаниями, включая документирование результатов.

Системное программное обеспечение управляющей ЭВМ (СПО): совокупность приложений и компонентов на управляющей ЭВМ, которая реализует пользовательский интерфейс, формирует команды и параметры управления, а также принимает результаты от ВПО по интерфейсам связи.

Тестовый сценарий на стороне управляющей ЭВМ: скрипт или программный модуль в составе СПО управляющей ЭВМ, который описывает последовательность шагов испытаний (тестирования), формирует команды и параметры для ВПО и обрабатывает полученные результаты. В составе ВПО вместо термина «сценарий» используется термин «алгоритм испытаний, реализованный в виде функций ВПО», и интерпретатор не применяется.

2. FormFrameWork: основы, преимущества, опыт применения и перспективы

2.1. FormFrameWork на фоне существующих решений

FormFrameWork – новая среди таких уже существующих программных платформ, как RTOS и сетевые стеки, Arduino-подобные фреймворки и др., предназначенных для ускорения разработки ВПО на микроконтроллерах и ПЛИС.

FFW является, по сути, заранее подготовленным программным «каркасом», а скорее даже системой, состоящей из библиотек, драйверов, шаблонов и правил, применение которой позволяет ограничить задачу программиста ВПО разработкой только прикладной логики устройства вместо того, чтобы писать ее с нуля. Разработчик существенно экономит время, поскольку дописывает только специфический для изделия код.

В отличие от «просто набора библиотек», платформа FFW навязывает определенный способ организации кода и инверсии управления: «каркас» вызывает код в predetermined моменты (коллбэки, задачи), а не наоборот.

2.2. Основные представления о FFW

Раскрывая основные представления о FormFrameWork, отметим, что это:

- Единый кросс-платформенный API, заменяющий SDK и библиотеки от разных производителей.
- Архитектура с четким разделением аппаратно-зависимого, системного и прикладного уровней, позволяющая быстро адаптировать платформу для аппаратных решений без переписывания всей системы, вплоть до пользовательских функций. При изменении «железа» достаточно внести изменения только в аппаратно-зависимый уровень (драйверы), а основная часть логики и системный код остаются без изменений.
- Система поддержки кооперативной многозадачности и механизма подписок с тем отличием, что предоставляет упрощенный доступ к общим ресурсам при сохранении детерминированности кода.
- Расширяемый комплект готовых модулей для работы с таймерами, безопасными обновлениями, энергонезависимой памятью, унифицированными интерфейсами периферии (GPIO, SPI) и др. для взаимодействия с командным процессором.
- Интуитивно понятный инструментарий для диагностики и отладки (библиотека FDebug).

FFW дает следующие преимущества:

- 1) экономит время на разработку;
- 2) снижает вероятность ошибок при миграции с одной элементной базы на другую, а также упрощает сопровождение систем;

- 3) позволяет повторно использовать системный код для разных аппаратных платформ;
- 4) структурирует управление изменениями в НИОКР, обеспечивает синхронизацию обновления и тестирования компонентов систем.
- 5) обеспечивает максимальную прослеживаемость обновлений ВПО в сложных системах с использованием десятков и сотен ПЛИС и МК;
- 6) встраивается в систему унификации и стандартизации НИОКР как типовой программный компонент и платформа вместо «зоопарка реализаций» в структуре стандартов разрозненных решений по каждому изделию и его составным частям.

2.3. Практические результаты

На уровне программ НИОКР компании ФОРМ платформа FFW фиксируется в структуре стандартов корпоративной технической политики. FFW на сегодняшний день уже многократно успешно используется в разработке командных процессоров для системных и измерительных модулей АТЕ FORMULA, выполненных по отечественному стандарту FormRXI, и показала высокую надежность, безопасность и возможность масштабирования.

В текущей конфигурации FFW поддерживает ВПО для следующих типов целевых аппаратных платформ (МК), входящих в спецификацию системных и измерительных модулей АТЕ FORMULA:

- MicroBlaze/ARMCortex-A9;
- Nios II;
- STM32.

2.4. Перспективы развития

Расширение поддержки аппаратных платформ, дальнейшая унификация библиотек, автоматизация конфигурации и сотрудничество с техническими комитетами для стандартизации практик импортонезависимой разработки.

3. Фундамент и предпосылки для создания отечественной программной платформы FFW

30-летний, без малого, опыт предприятия «ФОРМ» в области разработки АТЕ FORMULA показал, что переходу на импортонезависимые решения в области САПР объективно препятствует разнообразие используемых средств (Integrated Development Environment IDE, Software Development Kit SDK и пр.) от различных производителей микроконтроллеров и ПЛИС, преимущественно иностранных. При этом последствия ограничений доступа к зарубежным сайтам производителей для скачивания необходимых приложений преодолеваются при использовании платформы FFW, которая предоставляет разработчикам свой единый импортонезависимый кросс-платформенный API-интерфейс программирования приложений.

Кроме того, FFW обеспечивает разделение *системного* и *аппаратно-зависимого* уровней ВПО: системный код вызывает стандартные библиотечные функции, а аппаратно-зависимая специфика инкапсулирована в драйверах/адаптерах. Например, для интерфейсов типа SPI (Serial Peripheral Interface) можно отказаться от набора *частных* (принадлежащих производителям) драйверов в пользу *универсального* инструмента настройки и управления данным интерфейсом: задавать режимы Master/Slave, Duplex/Half-Duplex, CPOL/CPHA, SINGLE/DUAL/QUAD, SDR/DDR и все другие, необходимые в проектах.

Платформа FFW позволяет обходиться без использования возможностей и специфики RTOS (Real-Time Operating System – операционная система реального времени, OCPB), сохраняя при этом *детерминированность* и *неизменность* кода целевого приложения при замене аппаратных частей системы.

Следует отметить, что платформа FFW может поддерживать и ранее разработанные так называемые унаследованные (legacy) проекты, которые по разным причинам бывает целесообразно продолжать использовать, несмотря на их техническое устаревание и появление более современных решений.

4. Архитектурные принципы FFW

Проекты ВПО на основе платформы FFW реализованы в виде трехуровневой архитектуры:

- аппаратно-зависимый уровень;
- системный уровень;
- прикладной уровень.

Поскольку особенности реализации более низкого уровня для более высокого уровня скрыты и «неинтересны», разработчик может абстрагироваться от аппаратной реализации решений.

Основу работы FFW составляют принципы *кооперативной многозадачности* и *опциональной системы подписок*, которая рассматривается как средство достижения слабой связанности таких программных модулей, как библиотеки и драйверы.

Ключевым элементом FFW является функция `framework_cout`, которая выполняет диспетчеризацию задач. Все «заинтересованные» модули ВПО подписываются на ее вызов и исполняются кооперативно в основном цикле – так называемом *суперцикле*. Такой механизм устраняет жесткую зависимость разработки ВПО от RTOS и, тем самым, повышает ее предсказуемость.

Заметим, что при необходимости FFW может работать и поверх проприетарной ОС устройства, например в отдельном потоке RTOS.

Рассмотрим примеры, поясняющие ключевые опции FFW.

4.1. Синхронизация и таймеры: детерминизм без RTOS

Модуль *таймеров* платформы FFW обеспечивает обслуживание сотен программных таймеров (их количество ограничено лишь объ-

емом доступного ОЗУ) на одном аппаратном счетчике, что значительно экономит аппаратные ресурсы.

По достижении «назначенного времени» соответствующие запросы (callbacks) от таймеров становятся в очередь и вызываются ближайшей итерацией «суперцикла».

Дискретность таймеров ограничена 1 мс, что обеспечивает устойчивые временные характеристики работы ВПО.

4.2. Безопасные обновления и самовосстановление

Универсальный *модуль загрузчика* платформы FFW предназначен для безопасного дистанционного обновления ВПО без аппаратных программаторов с обеспечением целостности конфигурационных данных (по контрольной сумме CRC32).

Поддерживаются:

- разделение на основную и резервную области хранения конфигурационных данных;
- автоматический откат при ошибках.

4.3. Конфигурации и энергонезависимая память

Модуль `dev_setts` системного уровня платформы FFW управляет группами параметров, хранимыми в энергонезависимой флэш-памяти, и обеспечивает следующие операции:

- инициализация групп (ID, адрес, объем);
- запись/чтение;
- очистка.

Такой унифицированный способ хранения конфигурации и прав доступа, например паролей/режимов АТЕ RXI, позволяет не связывать прикладное ПО с конкретной микросхемой памяти.

```

1  | /*****
2  |  * @file main.c.
3  |  * @brief Пример использования GPIO. Выполняется обработка состояния кнопки (из прерывания).
4  |  * По спадающему фронту изменяется состояние светодиода.
5  |  * инициализацию устройств.
6  |  * @author a.tushentsov.
7  |  *****/
8  | #include "target.h"
9  | #include "System/framework.h"
10 | #include "System/supervisor.h"
11 | #include "System/sw_timer.h"
12 | #include "DRV/gpio.h"
13 | #include "mcu.h"
14 |
15 | static void gpio_isr_cb(u8 id, u8 gpio, gpio_state_e pin_state);
16 |
17 | /*****
18 |  * Функция инициализации FFW. Инициализирует подсистему тактирования, таймеры.
19 |  *****/
20 | void main() {
21 |     // Инициализация тактовой подсистемы, режимов энергосбережения.
22 |     supervisor_init();
23 |     // Инициализация таймеров.
24 |     sw_timer_init(NULL);
25 |     // Конфигурация вывода 14 порта А микроконтроллера как вход с подтяжкой к питанию. Задаем коллбэк на изменения уровня.
26 |     gpio_init(0, MCU_GPIO_P_A_14, GPIO_DIR_IN | GPIO_INT_EDGE_RISING | GPIO_INT_EDGE_FALLING | GPIO_PULL_UP, gpio_isr_cb);
27 |     // Конфигурация вывода 8 порта В микроконтроллера как выход.
28 |     gpio_init(0, MCU_GPIO_P_B_8, GPIO_DIR_OUT, NULL);
29 |     // Установка низкого уровня - соответствует состоянию светодиода "выключен"
30 |     gpio_set(0, MCU_GPIO_P_B_8, GPIO_STATE_LOW);
31 |
32 |     // Бесконечный цикл обхода подписчиков FFW
33 |     while(1) {
34 |         framework_cout();
35 |     }
36 | }
37 |
38 | /*****
39 |  * Коллбэк по изменению уровня на выводе 14 порта А микроконтроллера. Вызов в контексте прерывания!!!
40 |  *****/
41 | static void gpio_isr_cb(u8 id, u8 gpio, gpio_state_e pin_state) {
42 |     if (pin_state == GPIO_STATE_LOW)
43 |         gpio_set(0, MCU_GPIO_P_B_8, GPIO_STATE_TOGGLE);
44 | }

```

Рис. 1. Пример программы для микроконтроллера по управлению светодиодом через порт GPIO из прерывания

Драйвер *flash* системного уровня предоставляет такие типовые операции, как инициализация, *erase*, *read*, *write*, а также *callback*-модель, интегрированную в подписки суперцикла.

4.4. Унифицированные интерфейсы периферии

4.4.1. Интерфейс GPIO

Функции интерфейса реализованы в файле *gpio.c* FFW.

Через единый заголовок API поддерживаются стандартные операции:

- инициализации/деинициализации;
- установки/чтения состояния;
- альтернативные функции и базовая обработка прерываний (рис. 1).

4.4.2. Интерфейс SPI

Функции интерфейса реализованы в файле *spi.c* FFW.

Через единый заголовок API поддерживаются следующие опции настройки и управления:

- Master/Slave;
- CPOL/CPHA;
- Duplex (дуплекс/полудуплекс);
- скорость;
- формат кадра (7/8/16 бит);
- SDR/DDR;
- порядок бит MSB/LSB;
- логика CS;
- режим тактирования;
- ширина шины данных SINGLE/DUAL/QUAD и др.

Указанные режимы для конкретного интерфейса хранятся в отдельном заголовочном файле *.h*, представляющем собой таблицу (шаблон), данные которой заполнены из технического описания конкретной микросхемы. Для автоматизации разработки можно поручить заполнение таких шаблонов (на основе технического описания) какому-нибудь «продвинутому» искусственному интеллекту.

На рис. 2 представлен пример таблицы конфигурации интерфейсов SPI для нескольких устройств

4.5. Наблюдаемость: диагностика и отладка

Библиотека *FDebug* системного уровня API *fdebug.h* платформы FFW обеспечивает структурированную отладку и диагностику ВПО.

Поддерживаются:

- 1) задание уровней вывода;
- 2) отображение:

- событий инициализации/загрузки;
- состояния подсистем;
- этапов обмена;
- журнал драйверов;
- вывод сообщений в отладочные интерфейсы (UART, Ethernet).

Инициализация требует явной настройки уровней и обработчика событий; допускается их отключение компиляционным флагом.

5. Практика применения

Для облегчения освоения программной платформы FFW в состав ее эксплуатационной документации входят примеры с полным описанием. В частности, «Инструкция по применению модуля таймера», в которой приведен пример подключения FFW в САПР XILINX VITS 2019.2 с помощью микроконтроллера MicroBlaze.

Используются единый заголовочный файл конфигурации целевого приложения *target.h* и API таймеров, с помощью которых осуществляются:

- инициализация платформы;
- включение аппаратных прерываний таймера;
- запуск таймера;
- вывод в отладочный порт UART из *callback*-функции.

Для обновления конфигурации ПЛИС применен описанный выше загрузчик с формированием образов утилитой *parser.exe* и механизмом MultiBoot.

6. Результаты интеграции FFW – безопасность, надежность и устойчивость

По результатам расширенных *трехлетних* испытаний тестовой платформы FormRXI [1] и модульного АТЕ FORMULA в целом была подтверждена корректность работы ВПО из состава системных и измерительных модулей, разработанных с применением FFW и приложений, созданных на его основе.

Количество разработанных целевых модулей, длительность портирования на FFW, показатели устойчивости – все эти метрики формировались по данным внутренних протоколов, а также журналов библиотеки *FDebug*. На текущий момент мы считаем их достаточной валидацией безопасности и надежности FFW.

Платформа FFW позволила нам быстро разработать процессорную часть ВПО для 11 типов системных и измерительных модулей АТЕ FORMULA с реализацией их полного функционала согласно ТЗ и стандарту FormRXI. Причем, одновременно удалось понизить влияние типовых проблем с нехваткой ресурсов.

```

18 //**
19 * @brief Таблица соответствия устройств SPI и аппаратных интерфейсов.
20 * @attention Кол-во элементов в таблице spi_cfg_tbl должно соответствовать SPI_DEV_COUNT в target.h!
21 */
22 const spi_cfg_tbl_t spi_cfg_tbl[] = {
23     [SPI_MAIN_FLASH] = { .spi_id = 0, .cs_id = 1, .func = hw_spi_xil_func },
24     [SPI_AT25DF321_0] = { .spi_id = 0, .cs_id = (1 << 0), .func = hw_spi_form_func },
25     [SPI_AT25DF321_1] = { .spi_id = 1, .cs_id = (1 << 1), .func = hw_spi_form_func },
26     [SPI_CH0_EEPROM] = { .spi_id = 2, .cs_id = (1 << 3), .func = hw_spi_form_func },
27     [SPI_CH0_EXP] = { .spi_id = 3, .cs_id = (1 << 2), .func = hw_spi_form_func },
28     [SPI_CH1_EEPROM] = { .spi_id = 4, .cs_id = (1 << 5), .func = hw_spi_form_func },
29     [SPI_CH1_EXP] = { .spi_id = 5, .cs_id = (1 << 4), .func = hw_spi_form_func },
30     [SPI_CH2_EEPROM] = { .spi_id = 6, .cs_id = (1 << 7), .func = hw_spi_form_func },
31     [SPI_CH2_EXP] = { .spi_id = 7, .cs_id = (1 << 6), .func = hw_spi_form_func },
32     [SPI_CH3_EEPROM] = { .spi_id = 8, .cs_id = 1, .func = hw_spi_form_func },
33     [SPI_CH3_EXP] = { .spi_id = 9, .cs_id = 1, .func = hw_spi_form_func },
34     [SPI_SYNCCLOCK_G1] = { .spi_id = 10, .cs_id = 2, .func = hw_spi_form_func },
35     [SPI_SYNCCLOCK_G2] = { .spi_id = 10, .cs_id = 4, .func = hw_spi_form_func },
36     [SPI_SYNCCLOCK_SW] = { .spi_id = 10, .cs_id = 1, .func = NULL },
37     [SPI_SYNCCLOCK_CC] = { .spi_id = 10, .cs_id = 8, .func = hw_spi_form_func },
38
39 };

```

Рис. 2. Пример таблицы конфигурации интерфейсов SPI для нескольких устройств

Другие объекты, в ВПО которых успешно использована платформа FFW:

- стенды автоматизированных испытаний АТЕ (и его составных частей);
- автоматические средства для мониторинга параметров подсистем охлаждения и термостабилизации;
- автоматические средства управления авариями АТЕ, стенды для испытания подсистем питания, информационного обмена и синхронизации;
- специализированные роботы для прецизионной настройки и калибровки динамических параметров высокочастотного АТЕ.

7. Дорожная карта и стандартизация

План дальнейшего развития платформы FFW включает в себя:

- расширение набора поддерживаемых аппаратных платформ (МК/ПЛИС), в том числе МК32 АМУР и других отечественных микроконтроллеров и процессоров;
- унификацию набора библиотек;
- регламентацию выпуска безопасных обновлений.

В настоящее время зарегистрирована программа для ЭВМ FormFrameWork [2], и поданы документы на регистрацию товарного знака.

Предусмотрена автоматизация конфигурации системы – создание приложения «Конфигуратор FFW» с графическим пользовательским интерфейсом.

Публикация результатов и взаимодействие с ТК 165 «САПР электроники» будут направлены на распространение практик импорто-независимой разработки изделий электронной техники.

Выводы

Платформа FFW предлагает практичный и уже хорошо проверенный путь к импорто-независимой разработке ВПО, обеспечивающий: переносимость прикладного кода, детерминизм исполнения без RTOS, безопасные обновления и единый программный интерфейс для периферии.

Поддержка ПЛИС, глубоко и всесторонне подтвержденная работоспособность решений FFW в проекте создания модульного АТЕ

FORMULA по стандарту FormRXI дают все основания считать платформу FFW пригодной для масштабируемых проектов и серийных изделий.

Компания ФОРМ поддерживает актуальную редакцию технической документации FFW по скриптам и совместимости для корректного описания автоматизации команд и сценариев.

Для тех, кто будет использовать отечественный стандарт модульного тестеростроения FormRXI для создания заказных измерительных модулей для Тестеров, предлагаемая программная платформа FFW станет эффективным инженерным инструментом, предоставляющим готовые библиотеки командных процессоров, применение которых значительно сократит время разработки новых вариантов исполнения и конфигурирования отечественных модульных тестеров на платформе FormRXI.

Для разработчиков любых других систем с распределенным управлением платформа FFW станет тем системообразующим ядром проектов ВПО на микроконтроллерах и ПЛИС, которое не имеет границ для расширения номенклатуры библиотек (приложений).

В конечном счете, FFW может стать инструментом коллективного пользования инженеров, каждый из которых сможет, применяя его на практике, вносить свой вклад в ее развитие.

В перспективе нам представляется возможным создание сообщества типа Linux.

Мы продолжаем развивать платформу FFW так, чтобы она обладала высокой гибкостью и универсальностью. Эта особенность позволит применять ее не только в продукции компании ФОРМ, но и в тех отраслях, где требуется разработка ВПО на микроконтроллерах и ПЛИС. Благодаря модульной структуре и возможности расширения набора библиотек платформа можно адаптировать под конкретные задачи и потребности пользователей, обеспечив широкую функциональность, удобство внедрения, стабильность и импорто-независимость. ■

Литература

1. Тестер микросхем FORMULA RXI // <https://form.ru/module-system/rxi>.
2. Свидетельство о государственной регистрации программы для ЭВМ №2025686421 // [https://form.ru/upload/pdf/%D0%A1%D0%B2-%D0%B2%D0%BE_%D0%9F%D0%9E_%E2%84%962025686421_\(FFW\).pdf](https://form.ru/upload/pdf/%D0%A1%D0%B2-%D0%B2%D0%BE_%D0%9F%D0%9E_%E2%84%962025686421_(FFW).pdf).

СОБЫТИЯ, ЛЮДИ

ФОРМ приглашает на новую серию семинаров по контролю ЭКБ – 2026

Ждем 5 марта в Технопарке «Сколково» (Москва) на первый семинар специалистов, занятых контролем качества микросхем, реле и полупроводников на производстве, в дизайн-центрах, в испытательных лабораториях и на входном контроле.

В 2026 г. на наших практических встречах вы сможете:

- лично участвовать в проведении измерений параметров ЭКБ разных типов с использованием линейки тестеров FORMULA и тестовых решений Testbox;
- ознакомиться с полным циклом измерений ЭКБ, начиная с инженерной подготовки, редактирования измерительной программы и заканчивая выполнением контроля конкретной партии микросхем, полупроводников и реле;
- получить ответы от экспертов и обсудить практические кейсы.

Наша формула:

*интерактивная демонстрация + открытый диалог =
= практическая польза для вашей работы.*

Уже более 10 лет мы проводим семинары, объединяя интересы специалистов отрасли по актуальным вопросам входного контроля электропараметров ЭКБ. Наши мероприятия – это возможность вживую погрузиться в процесс тестирования.

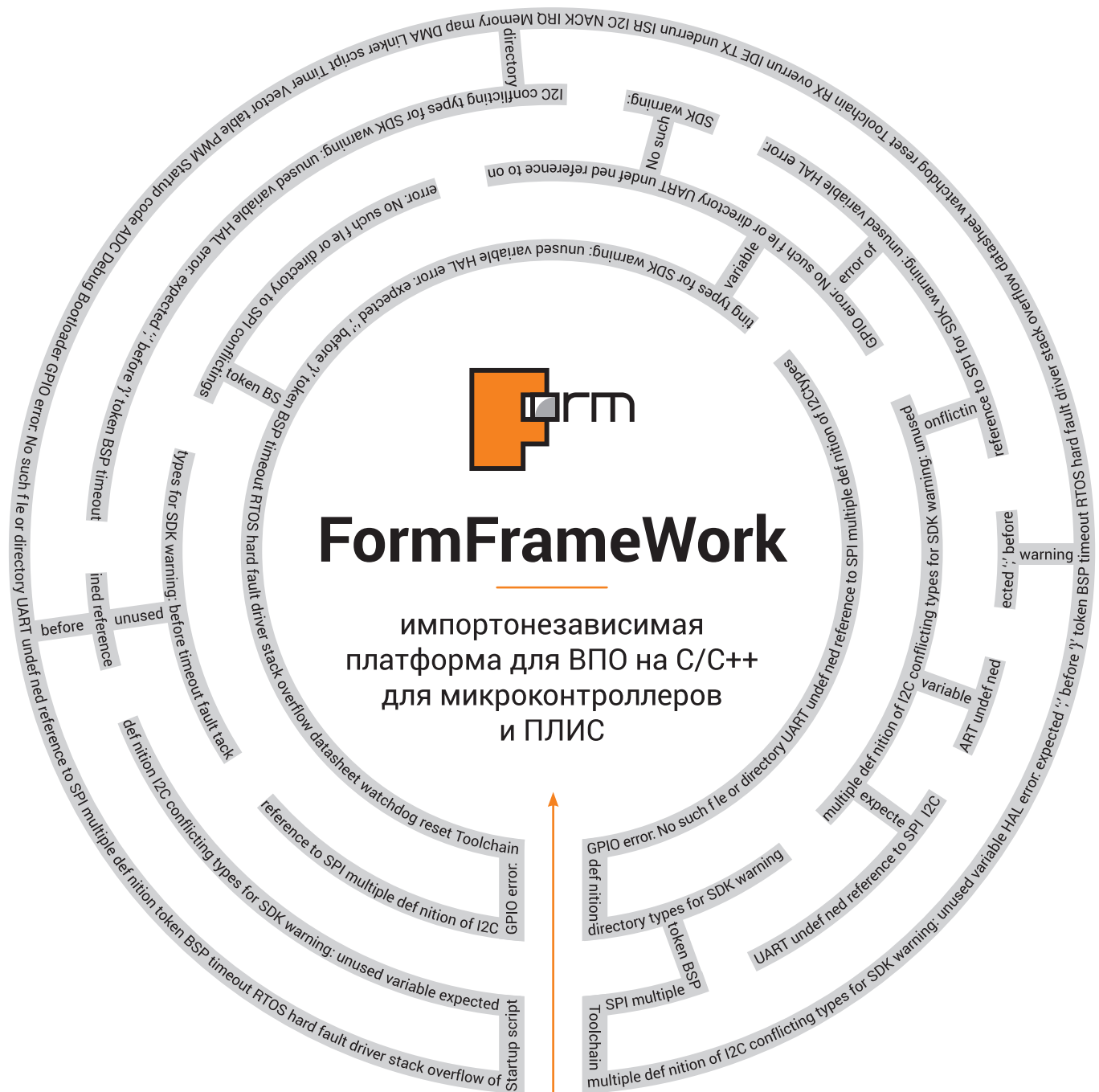


Следите за анонсами – программа и информация о регистрации скоро будут опубликованы в нашем Телеграм-канале FORMULA Микрoэлектроники и сайте www.form.ru.

Подробнее по тел.: +7 (495) 269-75-90.

ООО «ФОРМ» – российский разработчик и производитель систем измерения для тестирования ЭКБ с 1996 г.

Пишите логику, а не обвязку



- единый кроссплатформенный API вместо SDK производителей
- переносимость прикладной логики
- детерминированность без RTOS (суперцикл, кооперативная многозадачность)

Реклама

