Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б11-мм

Расширение функциональности Embox для применения в АСУ ТП

Пилюк Дмитрий Алексеевич

Отчёт по производственной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., Д. В. Луцив

Консультант:
ген. директор ООО «Ембокс» А. В. Бондарев

Санкт-Петербург
2025

Оглавление

Введение **3**

1. Постановка задачи **4**

2. Обзор **5**

2.1. ОСРВ Embox 5

2.2. АСУ ТП 5

2.3. Среда разработки 6

2.4. Eclipse 4diac 7

3. Реализация **8**

3.1. Добавление необходимых зависимостей в Embox 8

3.2. Добавления модуля 4diac FORTE 8

3.3. Добавление новых функциональных блоков 11

4. Эксперимент **17**

4.1. Условия 17

4.2. Программа 4diac IDE 17

4.3. Демо темплейт qemu 17

4.4. Демо темплейт qemu 18

4.5. Результат работы 18

Заключение **20**

Список литературы **21**

Введение

В наше время наблюдается развитие и повсеместное применение технологий автоматизации. Автоматизация позволяет значительно повысить производительность, снизить затраты, минимизировать влияние человеческого фактора и обеспечить выполнение сложных задач с высокой точностью. Внедрение автоматизированных систем управления стало ключевым драйвером прогресса в таких отраслях, как производство, транспорт, энергетика и медицина.

Одной из значимых областей применения автоматизации является управление технологическими процессами. В сложных промышленных системах с большим количеством взаимосвязанных элементов эффективность работы напрямую зависит от скорости обработки информации и точности принимаемых решений. Здесь на первый план выходят автоматизированные системы управления технологическими процессами (АСУ ТП). Они применяются в таких отраслях, как энергетика, нефтегазовая промышленность, машиностроение, химическая и пищевая промышленность.

Для выполнения данных задач широко используются операционные системы реального времени, которые способны гарантировать строгое соблюдение временных ограничений. Одной из таких ОСРВ является Embox, перспективное решение в данной области благодаря своим особенностям.

1. Постановка задачи

Целью работы является расширение возможностей ОСРВ Embox для использования в сфере АСУ ТП. Для её выполнения были поставлены следующие задачи:

1. Предоставить среду разработки, поддерживающую создание и редактирование программ стандарта IEC 61499 для Embox.
2. Добавить в Embox поддержку среды исполнения.
3. Провести эксперимент.

2. Обзор

2.1. ОСРВ Embox

Embox [2] — это конфигурируемая операционная система реального времени, предназначенная для встраиваемых систем с ограниченными ресурсами. Разработка начиналась в 2010 году на кафедре системного программирования Математико-механического факультета СПбГУ как студенческий проект.

Исходный код ОСРВ Embox представляет собой модульную систему, что позволяет гибко настраивать конечную сборку под конкретную задачу, с возможностью включить только минимально необходимые компоненты, либо получить ОС с богатой функциональностью. Эта модульность достигается за счёт использования системы сборки Mybuild.

Embox предоставляет POSIX-совместимость, благодаря чему позволяет портировать многие Linux приложения. Так же в состав проекта входит стандартная библиотека языка C.

Embox совмещает в себе возможность выполнять сложные функциональные задачи и является системой реального времени. Кроме того ОС имеет развитый сетевой стек, набор различных приложений, библиотек и при этом низкие аппаратные требования, поэтому хорошо подходит для использования в сфере АСУ ТП.

2.2. АСУ ТП

Автоматизированная система управления технологическим процессом (АСУ ТП) — система технических и программных средств, предназначенных для автоматизации управления технологическим процессом. В качестве аппаратной платформы используются специализированные ПЛК.

Программируемый логический контроллер (ПЛК) — цифровая электронная система, предназначенная для применения в производственной среде, которая использует программируемую память для внутреннего хранения ориентированных на потребителя инструкций по реализации

таких специальных функций, как логика, установление последовательности, согласование по времени, счёт и арифметические действия для контроля посредством цифрового или аналогового ввода/вывода данных различных видов машин или процессов.

Основными задачами в области АСУ ТП являются программирование ПЛК и организация их взаимодействия, что регламентируется стандартами IEC 31131 и IEC 61499.

IEC 31131 — набор стандартов МЭК для программируемых контроллеров.

IEC 31131-3 [3] — раздел международного стандарта IEC 31131, описывающий языки программирования для программируемых логических контроллеров. Включает языки LD, FBD, SFC, ST, IL.

IEC 61499 [4] — стандарт, расширяющий концепцию IEC 31131-3 для распределённых систем управления. Он основан на event-driven (событийной) архитектуре и функциональных блоках (FB), что позволяет создавать гибкие, масштабируемые и переиспользуемые решения для промышленной автоматизации. В отличие от IEC 61499, который ориентирован на цикличное выполнение программ, IEC 61499 поддерживает асинхронные события и распределённое выполнение задач, что делает его более подходящим для сложных, модульных и сетевых систем.

Так же в сфере АСУ ТП применяются различные стандарты взаимодействия, такие как CANopen, OPC-UA, Modbus, EtherCAT.

2.3. Среда разработки

Среди открытых решений поддерживающих языки стандарта IEC 31131-3 можно выделить:

- Beremiz — это интегрированная среда разработки для автоматизации машин. Это свободное программное обеспечение, соответствующее, среди прочих стандартов, стандарту IEC 31131. Так же предоставляет свою среду выполнения для программ. Поддерживает различные протоколы взаимодействия. Для генерации программ использует Matiec.

- OpenPLC Editor — форк от Beremiz. Так же предоставляет свою среду выполнения программ.

Среди открытых решений поддерживающих языки стандарта IEC 61499 можно выделить 4diac.

2.4. Eclipse 4diac

Eclipse 4diac [1] (Framework for Distributed Industrial Automation and Control) —это открытая среда разработки, предназначенная для создания распределённых систем промышленной автоматизации на основе стандарта IEC 61499. Проект разрабатывается как open-source решение и предоставляет инструменты для моделирования, симуляции и развёртывания управляющих приложений в промышленных системах.

Компоненты Eclipse 4diac:

- 4diac IDE — это интегрированная среда разработки (IDE), написанная на Java и основанная на платформе Eclipse. Она предоставляет инструменты для проектирования распределённых систем управления в соответствии со стандартом IEC 61499. В 4diac IDE можно создавать функциональные блоки (FB) и приложения, настраивать устройства и связи между ними, загружать (деплоить) готовые решения на устройства с 4diac FORTE.

- 4diac FORTE — это компактная и переносимая реализация среды исполнения (run-time environment) стандарта IEC 61499 на языке C++, предназначенная для выполнения сетей функциональных блоков (FB) на маломощных встраиваемых устройствах. FORTE работает поверх операционной системы.

3. Реализация

В данной работе реализована возможность запуска программ на Embox, созданных в 4diac IDE с добавление собственных функциональных блоков. Выделяются 3 основных раздела:

- Добавить в Embox необходимые зависимости для 4diac FORTE;
- добавить модуль 4diac FORTE в Embox;
- реализовать способ добавления новых функциональных блоков;
- добавить функциональные блоки в 4diac FORTE, управляющие светодиодами в Embox.

3.1. Добавление необходимых зависимостей в Embox

4diac FORTE написан на C++ и использует phtreads и libc. В Embox есть поддержка данных модулей, но некоторых функций не хватало. Для этого часть функций была реализована, а для другой части были использованы заглушки. Так же для использования C++ в Embox необходимо собирать GCC с окружением Embox, В ходе работы была обновлена версия GCC до 14.2.0[1].

3.2. Добавления модуля 4diac FORTE

Для сборки и конфигурирования Embox используется система сборки Mybuild. Каждый модуль в Embox описывает Mybuild файл. Так и для 4diac FORTE необходимо создать этот файл, его код представлен в листинге 1. В нем мы указываем модули Embox, которые необходимы для сборки 4diac FORTE, аннотации AutoCmd и CxxCmd задают правила сборки и делают так, чтобы в готовом образе Embox была команда forte. И @Build(stage=2, script="$(EXTERNAL_MAKE)") задает стадию сборки и указывает, что используется Makefile при сборке.

[1] https://github.com/embox/embox/pull/3579

Листинг 1: embox/project/4diac/forte/Mybuild

```
package project.fourdiac

@AutoCmd
@CxxCmd(name = "forte",
        help = "TODO"
    )
@BuildDepends(embox.lib.libstdcxx)
@Build(stage=2, script="$(EXTERNAL_MAKE)")
module forte {

        source "^BUILD/extbld/^MOD_PATH/install/forte.o"

    @NoRuntime depends embox.lib.libstdcxx
    @NoRuntime depends embox.compat.posix.pthread_stack
    @NoRuntime depends embox.compat.posix.util.nanosleep
    @NoRuntime depends embox.lib.leddrv
}
```

В Makefile описываются три цели Configure, Build и Install. Также файл содержит ссылку для скачивания исходного кода. Код представлен в листинге 2.

Листинг 2: embox/project/4diac/forte/Makefile

```
PKG_NAME := 4diac-forte
PKG_VER  := 2.0.1
PKG_SOURCES := \
https://github.com/eclipse-4diac/$(PKG_NAME)/archive/refs/tags/$(PKG_VE
PKG_MD5     := 6be3b6bd8b9276d8c22b8f38f6343515

FORTE_FLAGS = \
                -DFORTE_ARCHITECTURE=Posix \
                -DFORTE_COM_ETH=ON \
                -DFORTE_COM_FBDK=ON \
```

```
                -DFORTE_COM_LOCAL=ON \
                -DFORTE_TESTS=OFF \
                -DFORTE_TESTS_INC_DIRS="" \
                -DFORTE_TESTS_LINK_DIRS="" \
                -DFORTE_MODULE_CONVERT=ON \
                -DFORTE_MODULE_IEC61131=ON \
                -DFORTE_MODULE_UTILS=ON \
                -DCMAKE_C_COMPILER:PATH=$(EMBOX_GCC) \
                -DCMAKE_CXX_COMPILER:PATH=$(EMBOX_GXX) \
                -DCMAKE_SYSTEM_NAME:STRING=Generic \
                -DCMAKE_BUILD_TYPE:STRING=None \
                -DFORTE_EXTERNAL_MODULES_DIRECTORY= \
            $(PROJECT_DIR)/4diac/forte/Embox_FB_lib \


$(CONFIGURE):
        export EMBOX_GCC_LINK=full; \
        mkdir -p $(PKG_SOURCE_DIR)/bin/embox; \
        cd $(PKG_SOURCE_DIR)/bin/embox && ( \
                cmake $(FORTE_FLAGS) $(PKG_SOURCE_DIR) \
        )
        touch $@

$(BUILD):
        cd $(PKG_SOURCE_DIR)/bin/embox && ( \
                make \
        )
        touch $@

$(INSTALL):
        mv $(PKG_SOURCE_DIR)/bin/embox/src/forte \
            $(PKG_INSTALL_DIR)/forte.o
        touch $@
```

3.3. Добавление новых функциональных блоков

В прошлой секции 4diac FORTE был сконфигурирован так (опция -DFORTE_EXTERNAL_MODULES_DIRECTORY = $(PROJECT_DIR)/4diac/forte/Embox_FB_lib), чтобы брал все дополнительные модули из директории /path/to/embox/project/4diac/Embox_FB_lib. Для этого необходимо по данному пути создать модуль, который будет содержать .cpp и .h файл для каждого функционального блока и CMakeLists.txt, в котором указаны все необходимые файлы для включения и название модуля.

Также был добавлен модуль управления светодиодами в Embox. Для этого в 4diac IDE были созданы интерфейсы функциональных блоков для включение светодиода рис. 1 и для выключения рис. 2.

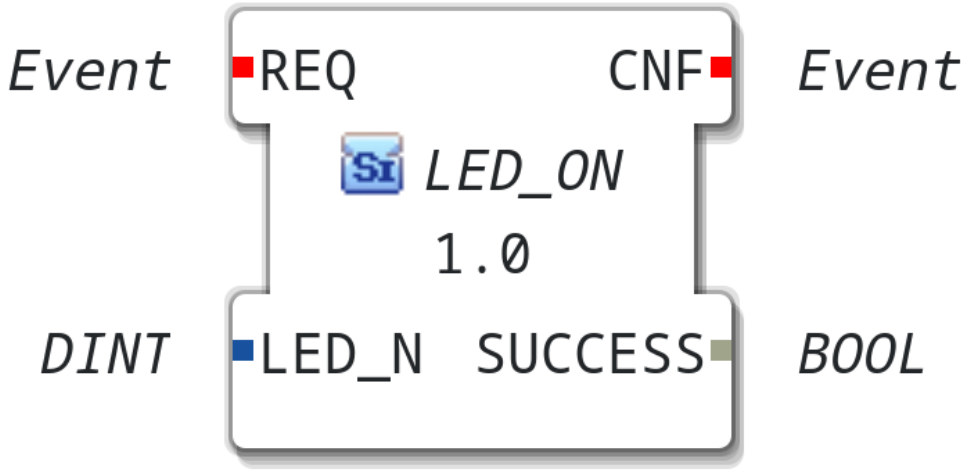

Рис. 1: LED_ON

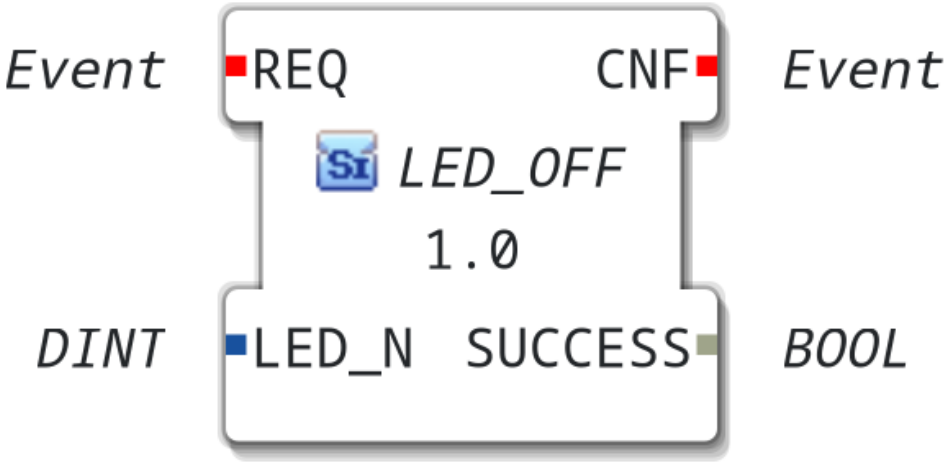

Рис. 2: LED_OFF

А в Embox была добавлена реализация поведения данных функциональных блоков. Код для LED_ON представлен в листинге 3 и 4. Для LED_OFF аналогично.

Листинг 3: embox/project/4diac/Embox_FB_Lib/LED_ON.h

```
#ifndef _LED_ON_H_
#define _LED_ON_H_

#include "funcbloc.h"
#include "forte_bool.h"
#include "forte_dint.h"
#include "forte_array_at.h"

#include <lib/leddrv.h>

class FORTE_LED_ON: public CFunctionBlock {
  DECLARE_FIRMWARE_FB(FORTE_LED_ON)

private:
  static const CStringDictionary::TStringId scm_anDataInputNames[];
  static const CStringDictionary::TStringId scm_anDataInputTypeIds[];
```

```
  static const CStringDictionary::TStringId scm_anDataOutputNames[];
  static const CStringDictionary::TStringId scm_anDataOutputTypeIds[];

  static const TEventID scm_nEventREQID = 0;

  static const TForteInt16 scm_anEIWithIndexes[];
  static const CStringDictionary::TStringId scm_anEventInputNames[];

  static const TEventID scm_nEventCNFID = 0;

  static const TForteInt16 scm_anEOWithIndexes[];
  static const CStringDictionary::TStringId scm_anEventOutputNames[];


  static const SFBInterfaceSpec scm_stFBInterfaceSpec;

  CIEC_DINT &st_LED_N() {
    return *static_cast<CIEC_DINT*>(getDI(0));
  }

  CIEC_BOOL &st_SUCCESS() {
    return *static_cast<CIEC_BOOL*>(getDO(0));
  }


  FORTE_FB_DATA_ARRAY(1, 1, 1, 0);

  void executeEvent(int pa_nEIID);

public:
   FORTE_LED_ON(const CStringDictionary::TStringId pa_nInstanceNameId,
                CResource *pa_poSrcRes) :
       CFunctionBlock( pa_poSrcRes,
```

```
                    &scm_stFBInterfaceSpec,
                    pa_nInstanceNameId,
                    m_anFBConnData,
                    m_anFBVarsData) {
   };

  virtual ~FORTE_LED_ON() = default;
};

#endif // _LED_ON_H_
```

Листинг 4: embox/project/4diac/Embox_FB_lib/LED_ON.cpp

```
#ifndef _LED_ON_H_
#define _LED_ON_H_

#include "funcbloc.h"
#include "forte_bool.h"
#include "forte_dint.h"
#include "forte_array_at.h"

#include <lib/leddrv.h>

class FORTE_LED_ON: public CFunctionBlock {
  DECLARE_FIRMWARE_FB(FORTE_LED_ON)

private:
  static const CStringDictionary::TStringId scm_anDataInputNames[];
  static const CStringDictionary::TStringId scm_anDataInputTypeIds[];

  static const CStringDictionary::TStringId scm_anDataOutputNames[];
  static const CStringDictionary::TStringId scm_anDataOutputTypeIds[];

  static const TEventID scm_nEventREQID = 0;
```

```
  static const TForteInt16 scm_anEIWithIndexes[];
  static const CStringDictionary::TStringId scm_anEventInputNames[];

  static const TEventID scm_nEventCNFID = 0;

  static const TForteInt16 scm_anEOWithIndexes[];
  static const CStringDictionary::TStringId scm_anEventOutputNames[];


  static const SFBInterfaceSpec scm_stFBInterfaceSpec;

  CIEC_DINT &st_LED_N() {
    return *static_cast<CIEC_DINT*>(getDI(0));
  }

  CIEC_BOOL &st_SUCCESS() {
    return *static_cast<CIEC_BOOL*>(getDO(0));
  }


  FORTE_FB_DATA_ARRAY(1, 1, 1, 0);

  void executeEvent(int pa_nEIID);

public:
   FORTE_LED_ON(const CStringDictionary::TStringId pa_nInstanceNameId,
                CResource *pa_poSrcRes) :
       CFunctionBlock( pa_poSrcRes,
                       &scm_stFBInterfaceSpec,
                       pa_nInstanceNameId,
                       m_anFBConnData,
                       m_anFBVarsData) {
```

```
    };

  virtual ~FORTE_LED_ON() = default;
};

#endif // _LED_ON_H_
```

4. Эксперимент

4.1. Условия

В качестве эксперимента будет создана программа в 4diac IDE, которая будет мигать светодиодом на Embox, запущеном на qemu, а затем на плате. Для этого необходимо:

- реализовать программу в 4diac IDE;
- создать темплейты в Embox с полученной программой.

4.2. Программа 4diac IDE

Программа для управления светодиодом представлена на рис. 3. На целевое устройство программу можно загрузить подключившись по сети или импортировать fboot файл и загрузить его в файловую систему устройства. В данном случае воспользуемся вторым способом. Полученный файл назван Blink_Embox_qemu.fboot будет помещен в rootfs для созданных темплейтов.

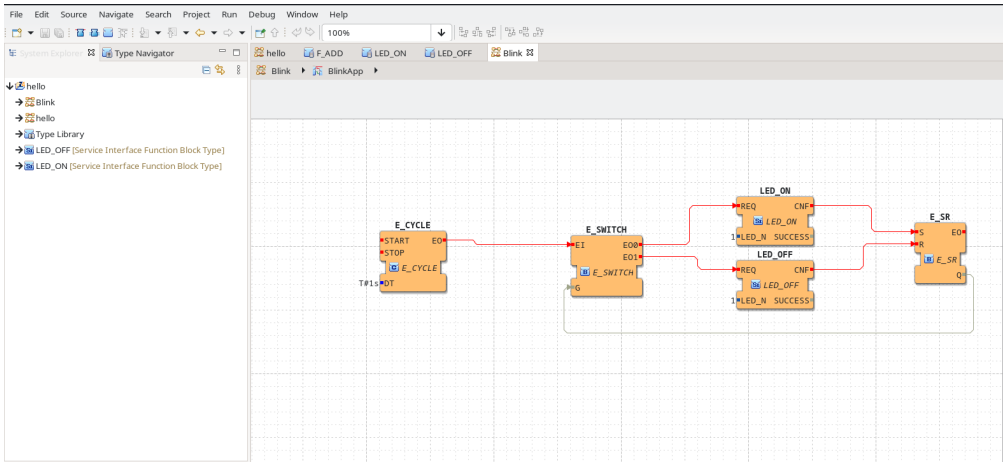

Рис. 3: Программа в 4diac IDE.

4.3. Демо темплейт qemu

В директории embox/project/4diac/templates создадим темплейт arm_qemu добавим в него сгенерированную программу. Команды для сборки и за-

пуска представлены в листинге 5.

Листинг 5: Инструкция по сборке и запуску.

```
// В директории embox
make confload-project/4diac/arm_qemu
make
./scripts/qemu/autoqemu

// В tish в эмуляторе для запуска программы
forte -f Blink\_Embox\_qemu.fboot
```

4.4. Демо темплейт qemu

В директории embox/project/4diac/templates создадим темплейт nucleo_f429zi, добавим в него сгенерированную программу. Команды для сборки представлен в листинге 6. Инструкцию по загрузке и запуску можно посмотреть на wiki[2]

Листинг 6: Инструкция по сборке.

```
// В директории embox
make confload-project/4diac/nucleo_f429zi
make
```

4.5. Результат работы

Результат работы на qemu представлен на рис. 4. Созданная 4DIAC IDE программа успешно запустилась и меняет состояние светодиода. На плате тоже успешно работает.

[2] https://github.com/embox/embox/wiki/NUCLEO-F429ZI

```
>netmanager
>tish
root@embox:/#forte -f Blink_Embox_qemu.fboot
INFO: T#12509014260: Using provided bootfile location set in the command line: Blink_Embox_qemu.fboot
INFO: T#12512470140: Boot file Blink_Embox_qemu.fboot opened
INFO: T#12521066200: FORTE is up and running
INFO: T#12523141340: Bootfile correctly loaded
INFO: T#12524372640: Closing bootfile
[info] (leddrv_stub) LED[0] := 1
[info] (leddrv_stub) LED[0] := 0
[info] (leddrv_stub) LED[0] := 1
[info] (leddrv_stub) LED[0] := 0
[info] (leddrv_stub) LED[0] := 1
[info] (leddrv_stub) LED[0] := 0
[info] (leddrv_stub) LED[0] := 1
[info] (leddrv_stub) LED[0] := 0
[info] (leddrv_stub) LED[0] := 1
```

Рис. 4: Результат работы.

Заключение

Данная работа направлена на расширение возможностей ОСРВ Embox для использования в сфере АСУ ТП.

Для этого было сделано:

1. Выбран проект Eclipse 4diac для создания программ. Реализована возможность добавления собственных функциональных блоков в Embox.

2. Добавлена среда исполнения 4diac FORTE.

3. Проведен эксперимент.

С кодом проекта можно ознакомится в репозиторие embox[3].

[3]Код проекта Embox: https://github.com/embox/embox
Пул реквест 1: https://github.com/embox/embox/pull/3579
Пул реквест 2: https://github.com/embox/embox/pull/3603
Пул реквест 3: https://github.com/embox/embox/pull/3609
Пул реквест 4: https://github.com/embox/embox/pull/3612
Пул реквест 5: https://github.com/embox/embox/pull/3630

Список литературы

[1] 4diac. 4diac. — URL: https://eclipse.dev/4diac/ (дата обращения: 15 июня 2025 г.).

[2] Embox. Wiki. — URL: https://github.com/embox/embox/wiki (дата обращения: 25 октября 2024 г.).

[3] Wikipedia. IEC_61131-3. — URL: https://en.wikipedia.org/wiki/IEC_61131-3 (дата обращения: 25 октября 2024 г.).

[4] Wikipedia. IEC_61499. — URL: https://ru.wikipedia.org/wiki/IEC_61499 (дата обращения: 14 июня 2025 г.).