Разработка и исследование методов и алгоритмов синтеза программ управления для гетерогенных микроконтроллерных систем

Аспирант: Большаков О. С.

Руководитель: к.ф.-м.н. Шаров В. Г.

Встраиваемые системы

Классификация встраиваемых систем (по Платунову сделать ссылку):

–промышленные ПК

–программируемые логические контроллеры (ПЛК, PLC) и программируемые контроллеры автоматизации (ПАК, PAC)

–мобильные и интернет-устройства (смартфоны и планшеты)

–контроллерные (Fieldbus) и сенсорные сети

–**микроконтроллерные системы**

–сигнальные процессоры (DSP)

–программируемая логика — ПЛИС (PLD, FPGA)

–заказные СБИС (ASIC, ASIP, SoC, Network on Chip – NoC).

Микроконтроллерные системы

Задачи, решаемые МК-системами:

- Сбор данных с датчиков
- Управление устройствами
- Передача данных другим вычислителям

Сферы применения:

- быт (бытовая техника, мобильные системы, системы умного дома)
- биомедицина
- робототехника
- производство (киберфизические системы)
- мониторинг (сенсорные сети)
- опытно-конструкторская деятельность (прототипирование новых устройств)

Особенности МК-систем

- Гетерогенность, специализация оборудования под задачи
- Ограниченное количество ресурсов (память – на 5-6 порядков меньше ПК, частота – на 3 порядка);
- Использование для связи устройств различных интерфейсов
- Ограничения по энергоэффективности (часто - автономное питание)
- Ограниченные габариты
- Ограниченный вес

Проблемы разработки программ для микроконтроллерных систем

Проблемы:

- Низкий уровень представления (абстракции) системы (программирование в логике битов памяти)
- Непортируемость программ
- Отсутствие унифицированных библиотек доступа к функциональным блокам оборудования
- Сложности реализации параллелизма, отсутствие унифицированных библиотек планировщиков параллельных процессов
- Сложности реализации низкоуровневых протоколов взаимодействия устройств
- Архаичность используемых низкоуровневых языков программирования

Синтез программ для МК-систем

Автоматизированный синтез программ – получение низкоуровнего кода программ (реализации) по описанной высокоуровневой спецификации с возможностью задания параметров синтеза.

Типовые подходы к решению задач синтеза:

- Генераторы инициализационного кода: Atmel Start, STM32CubeMX, Grace, CodeVisionAVR;
- Инструменты синтеза программ по моделям конечных автоматов и потоков данных (SCADE, Matlab, IAR Embedded workbench и т.д.)
  - разрыв между моделью представления программы на высоком уровне и моделью представления программы на уровне С
  - неполный синтез: сгенерированный код необходимо вручную привязывать к оборудованию
- Синтез по императивным графическим моделям (блок-схемы FlowCode).

Проблемы:

– Отсутствует возможность синтеза алгоритмов взаимодействия распределенных модулей
– Отсутствует поддержка кооперативной многозадачности параллельных процессов.

Цели и задачи

Цель исследования:

Разработка моделей и алгоритмов, обеспечивающих автоматизированный синтез программ для встраиваемых микроконтроллерных систем на основе высокоуровневой спецификации

Задачи:

1) Разработка обобщенной модели представления программ для микроконтроллеров на высоком уровне абстракции.

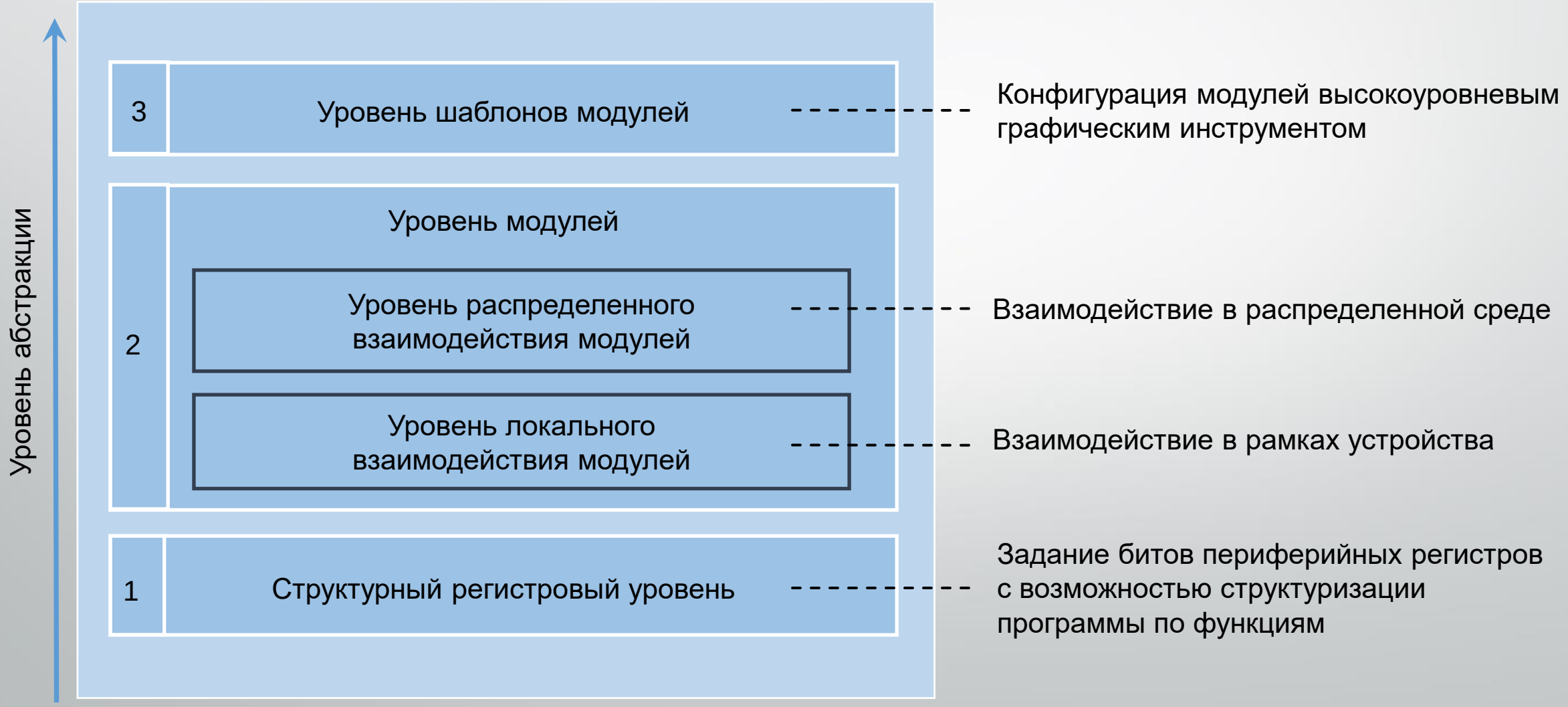
2) Разработка алгоритмов, обеспечивающих автоматический синтез системы с параллельным исполнением задач

3) Разработка алгоритмов, обеспечивающих автоматический синтез системы с взаимодействием программ в распределенной среде

4) Разработка языка программирования как инструмента высокоуровневой разработки встраиваемых микроконтроллерных систем

5) Создание инструментальной среды эффективной разработки программ для микроконтроллерных систем на основе предложенных моделей и алгоритмов

Уровни абстракции в представлении программ



Концептуальная модель представления программ

1. Уровень шаблонов

Част. специализация

$\{p\} - \{p_s\}$

Шаблон интерфейса

Параметры: P1, P2, …, Pv

Реализация

Шаблон модуля

Параметры: P1, P2, …, Pw

2. Уровень модулей

Инстанциация

Интерфейс

Реализация

Инстанциация

Модуль

Интерфейсный Модуль

Задача
- название
- сообщения
- события
- функции
- структ. данных

Поток управления

Пакет
- название
- функции
- события
- структ. данных

$$Templs = \{templ^i\}, i \in \{1..n\}, n \geq 0$$

$$templ^i = \langle name_{templ^i}, Param_1, \dots, Param_m, Code_{templ^i} \rangle$$

$$Finst^i : Params_{templ^i} \rightarrow Modules_{inst}$$

$$Params_{templ^i} = Param_1 \times \cdots \times Param_m$$

1. Структурный регистровый уровень

Микроконтроллер

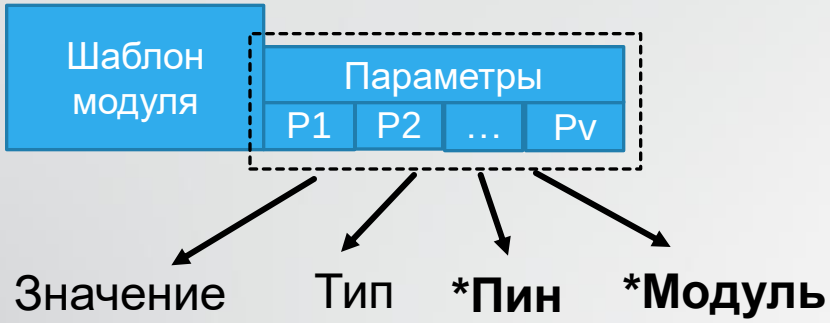
R1 R2 … R2

Запись

Чтение

Тело функции
- Параметры
- Структуры данных
- Инструкции

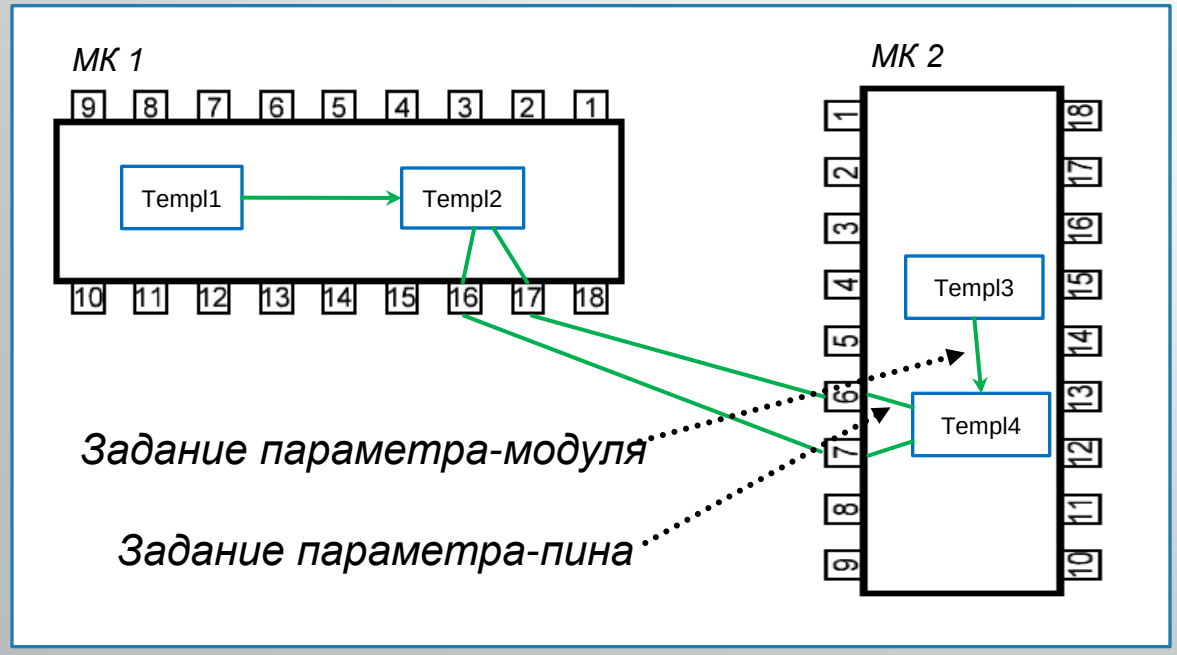
Шаблоны модулей и шаблоны интерфейсов



Концепция **шаблонов модулей** реализует принцип параметрического полиморфизма и основана на шаблонах классов C++. Отличия:

- Добавлены новые виды параметров: модуль и пин.
- При описании модуля-параметра возможно указать интерфейс модуля-параметра.
- При описании типа-параметра есть возможность ограничить параметр списком возможных значений типов.

Это позволяет параметризировать шаблоны в графическом виде



Конфигурация параметров в графическом виде

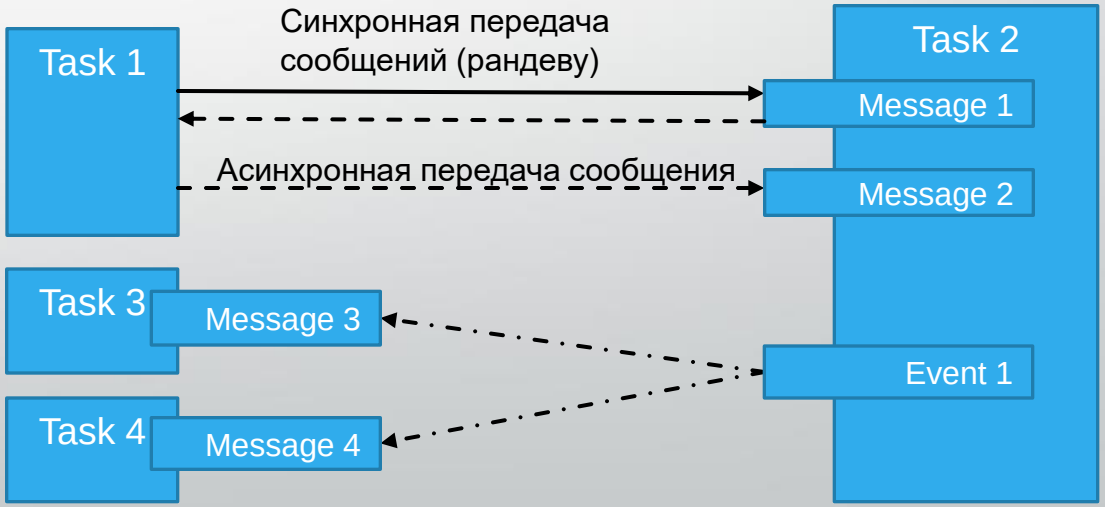
Концепция **шаблонов интерфейсов** позволяет создать методику разработки программного кода, при которой производится выделение кроссплатформенных и некроссплатформенных элементов интерфейса библиотечных модулей

Модули

- Модули выделяются разработчиком по функциональным признакам на этапе проектирования системы
- Выделяется два вида модулей: **задачи и пакеты**. Задачи содержат поток управления, функции пакетов вызываются в потоках управления задач.
- Модули могут взаимодействовать друг с другом за счет отправки сообщений (задачам) и вызова функций пакетов. Также имеется механизм параметризированных событий, позволяющий выполнять нескольким модулям подписку на некоторое событие, происходящее в другом модуле.
- Выделяется особый тип модулей – **интерфейсные**, через которые происходит взаимодействие между модулями, находящимися в разных устройствах. Данный тип модулей реализует унифицированный API и позволяет осуществлять синтез алгоритмов взаимодействия модулей между устройствами

Особенности задач в предлагаемой модели:

- Между задачами возможно синхронное и асинхронное взаимодействие
- В структуру задачи добавлен механизм событий, что позволяет задачам быть инициаторами событий для подписчиков, что существенно расширяет возможности при разработке библиотеки компонентов.



Особенности языка Embeddecy

Преимущества языка	Какими средствами языка обеспечивается
Совместимость с языком C	Синтаксис и семантика программ основана на языке C, у пользователя есть возможность улучшить характеристики языка, отключая совместимость с некоторыми концепциями языка C в части строгости синтаксиса
Поддержка параллельного программирования	Описание задач с выделением потока управления, конструкции запуска задач, приостановки, продолжения исполнения, задержки, синхронной и асинхронной отправки сообщений, принятия сообщений
Поддержка распределенного взаимодействия	Интерфейсные модули, конструкция **via** для взаимодействия через интерфейсный модуль
Поддержка шаблонов	Описание шаблонов интерфейсов и модулей с параметрами трех типов: тип, значение, модуль. Возможность перечисления возможных типов для параметра-типа. Возможность частичной специализации параметров-интерфейсов.
Средства синтаксической выразительности	Пространства имен, события, делегаты, анонимные функции, удобная работа с битами, определение типов с произвольным диапазоном возможных значений и т.д.

Конструкции параллельного программирования и передачи сообщений в Embeddecy

Тип конструкций	Примеры конструкций

Описание задачи	task myTask {
	void func1() {} // функция в задаче
	
	body { // тело задачи, начало потока управления
	func1();
	}
	}

Команды по работе с задачами	task1.start(); // запустить
	task1.stop(); // остановить
	task1.pause(); // приостановить
	task1.continue(); // продолжить исполнение
	task1.delay_ms(1000); // задержка в мс

Конструкции для отправки и принятия сообщений	send task2.message1(3, 4); // послать сообщение асинхронно
	task2.message1(3, 4); // послать сообщение синхронно
	hasmessage(message2); // пришло ли сообщение message2
	accept message2(int x, float y) do { … }; // принять сообщение message2
	accept message3 timeout 5s or {…}; // принять сообщение message3 с таймаутом
	send task2.message1(3, 4) via spi1; // послать сообщение через интерфейсный модуль

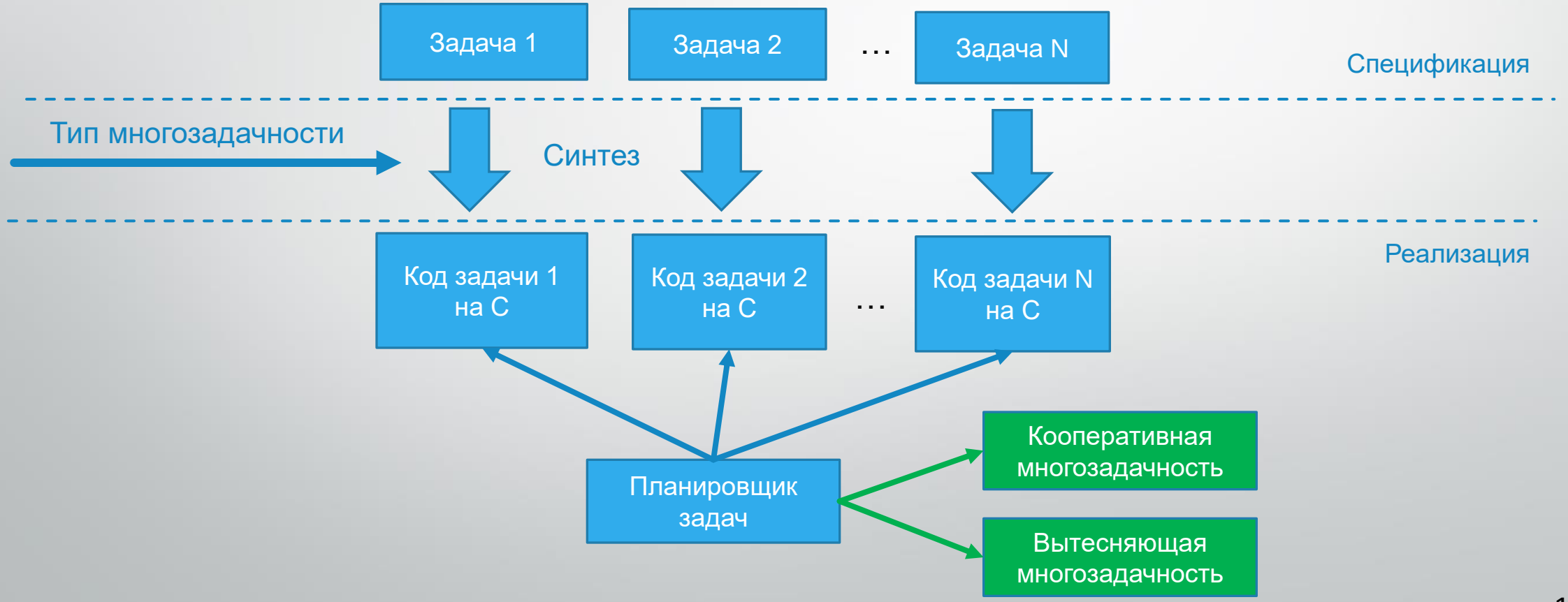
Сравнение Embeddecy с C, C++, Ada

	C	C++	Ada	Embeddecy
Типизация	Слабая статическая	Слабая статическая, средняя динамическая	Сильная статическая	Слабая + сильная статическая, утиная
Параметрический полиморфизм (поддержка шаблонов)	-	+-	-	+
Степень связности модулей	Слабые языковые механизмы	Присутствуют некоторые механизмы	Повышенная связность, в том числе связность параллельных модулей	Высокая связность задач, на пакеты ограничений нет
Языковые средства обеспечения паралл. програм-ия	-	-	+	+
Делегирование (подписка на сообщения)	-	-	-	Поддерживается, имеется удобный синтаксис с использованием анонимных функций
Абстрагирование от коммуникационных протоколов	-	-	-	+

Синтез программного кода по обеспечению псевдопараллельного исполнения параллельных модулей

Синтез низкоуровневого кода на языке С, реализующего параллельное исполнение задач:

1) с кооперативным типом многозадачности
2) с вытесняющим типом многозадачности



Синтез программного кода, обеспечивающего параллельное исполнение модулей (кооперативная многозадачность)

Преимущества кооперативного типа многозадачности:

1)Отсутствие необходимости синхронизировать доступ к разделяемым данным

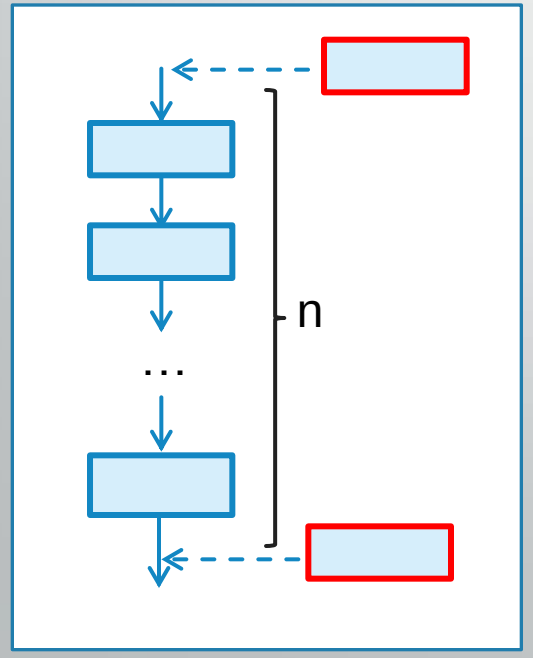
2)При отсутствии вложенных вызовов функций из тела задачи отсутствие необходимости сохранять стек локальных переменных.

Проблема: необходимость расстановки в коде инструкций передачи управления планировщику

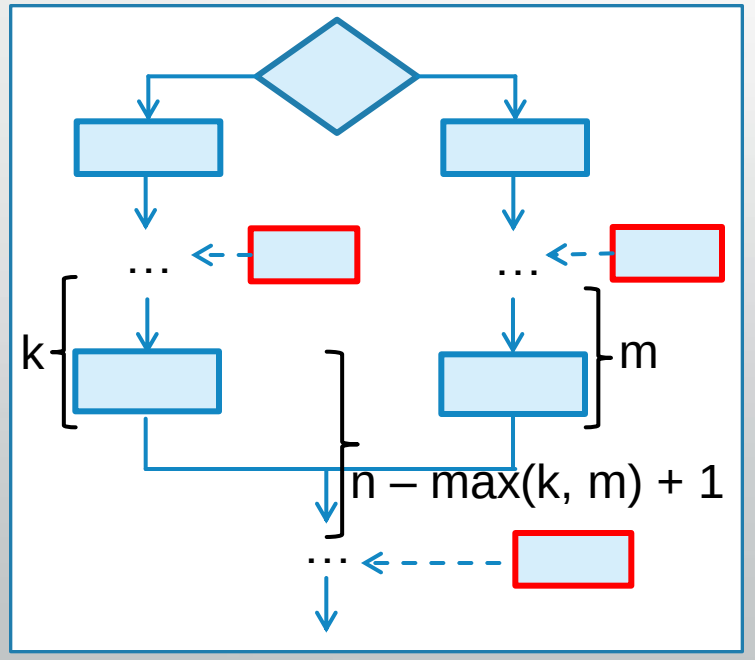
Решение: автоматическая расстановка конструкций при генерации кода

Параметр генерации: n - допустимое количество низкоуровневых инструкций между инструкциями по передаче управления планировщику

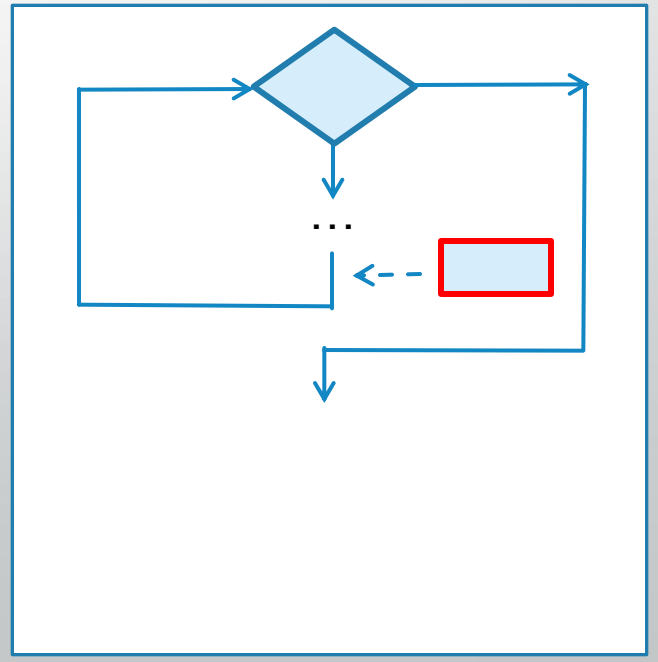
Последовательные фрагменты



Ветвления



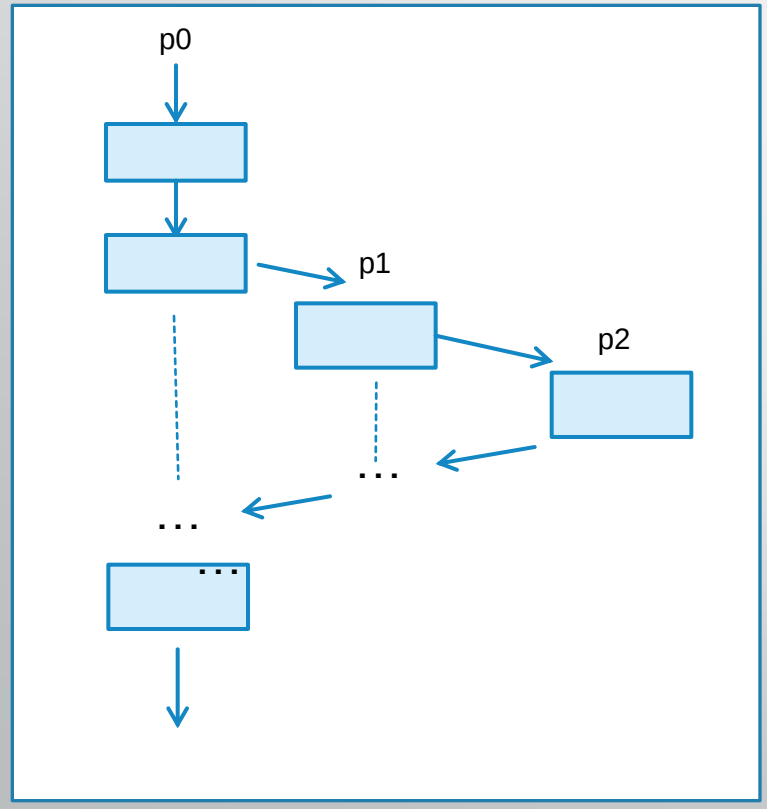
Циклы



Синтез программного кода, обеспечивающего параллельное исполнение модулей (вытесняющая многозадачность)

Проблема: определение размера стека для сохранения контекста

Решение: размер стека можно определить по максимальной вложенности вызовов на уровне описания инструкций.



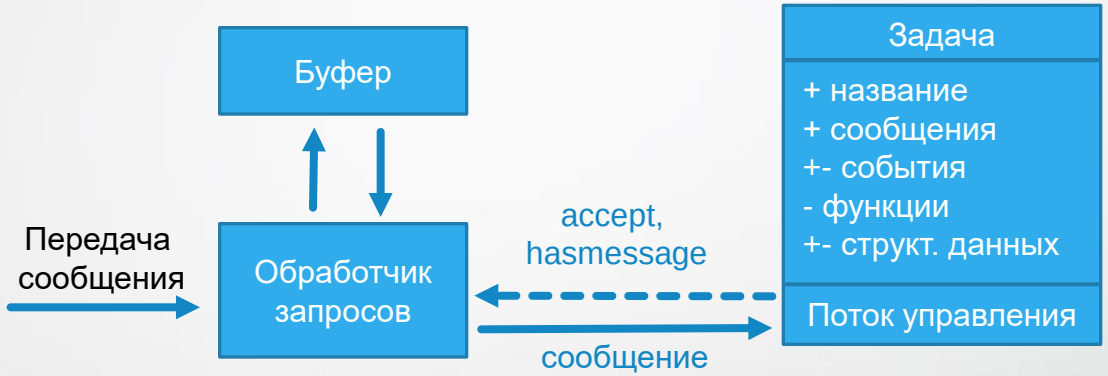
Алгоритм: обход в глубину дерева вызовов на уровне описания инструкций. Каждый вызов функции подразумевает увеличение размера стека для контекста на величину стека для этой функции.

StackMax = Ssize(p0) + Ssize(p1) + Ssize(p2)

Генерация программного кода для обмена сообщениями между программными модулями одного устройства

Вызов функции пакета аналогичен вызову обычной функции.

Задаче можно передать сообщение, которое принимается буфером и в ее потоке управления принимается по ее желанию.



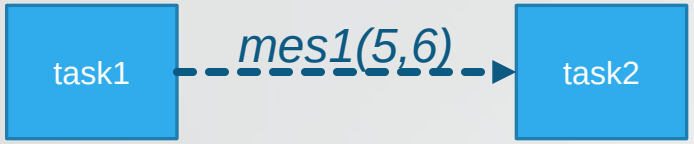
Задачи генерации:

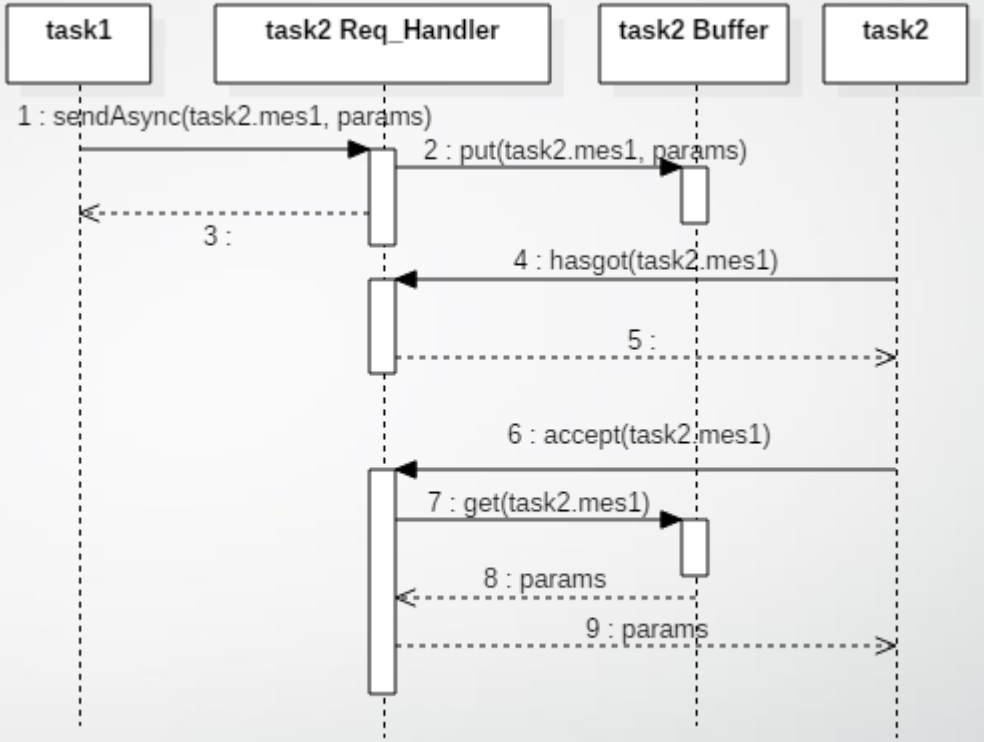
1) Для каждой задачи сгенерировать обработчики запросов, включающие функции приема сообщений + буферы (структуры данных типа «FIFO»).

2) Сгенерировать код отправляющего модуля по передаче сообщения и ожиданию ответа (если взаимодействие синхронное).

3) Сгенерировать код принимающего модуля по принятию сообщения и выдаче ответа о завершении принятия (при синхронном взаимодействии).

Синтез алгоритма асинхронного взаимодействия в рамках одного устройства



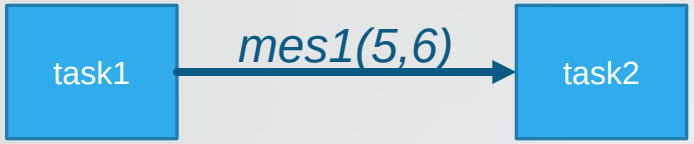


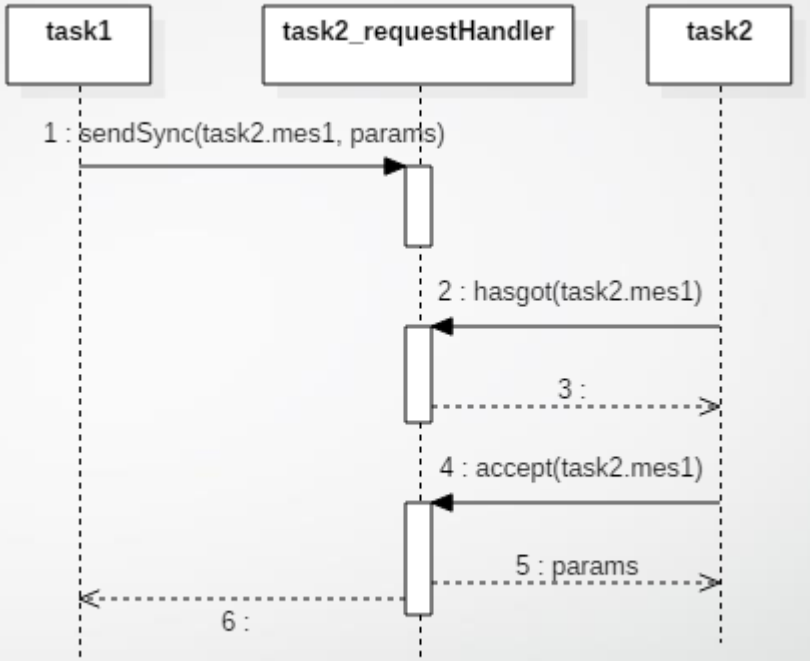
При синтезе асинхронного взаимодействия, сообщение посылается обработчику запросов асинхронно и кладется в буфер задачи. Принимающая задача забирает сообщение из буфера через обработчик запросов.

	task1:	task2:
Embeddecy	send task2.mes1(5, 6);	int x, y, a; accept mes1(x, y) do { a = x + y; … }

Генерируемый C	task2_reqhandler_sendAsync(task2_mes1, 5, 6);	int x, y;
		while (! task2_reqhandler_hasgot(task2_mes1);
		task2_reqhandler_accept (task2_mes1, &x, &y);
		a = x + y;
		…

Синтез алгоритма синхронного взаимодействия в рамках одного устройства





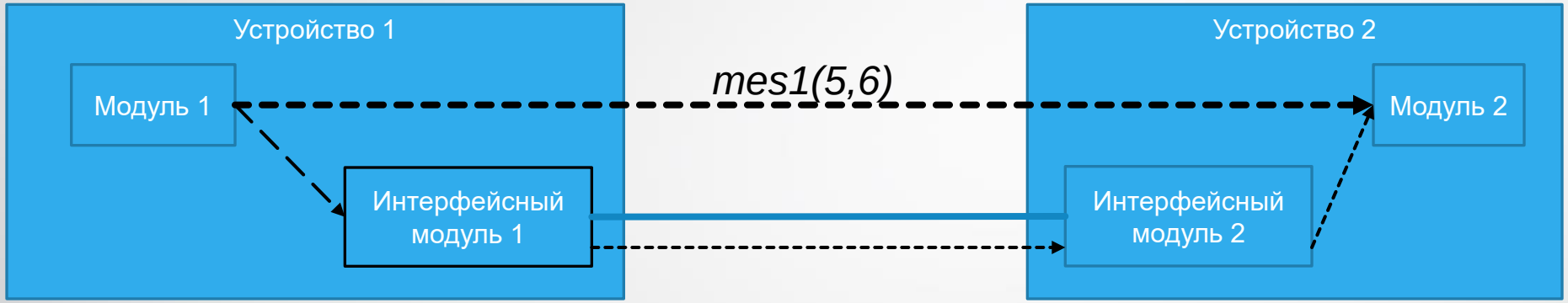
При синтезе синхронного взаимодействия, сообщение посылается обработчику запросов синхронно и ожидается принятие целевой задачей. Сообщение в буфер при этом не кладется.

	task1:	task2:
Embeddecy	task2.mes1(5, 6);	int x, y, a; accept mes1(x, y) do { a = x + y; … }

Генерируемый C	task2_reqHandler_sendSync(task2_mes1, 5, 6);	int x, y;
		while (! task2_reqhandler_hasgot(task2_mes1);
		task2_reqhandler_accept(task2_mes1, &x, &y);
		a = x + y;
		…

Генерация программного кода для обмена сообщениями между программными модулями разных устройств

Модули между разными устройствами напрямую передают друг другу сообщения, указывая, через какой интерфейсный модуль необходимо произвести взаимодействие



Задачи:

1) Разработать унифицированную архитектуру интерфейсных модулей и унифицированное API.

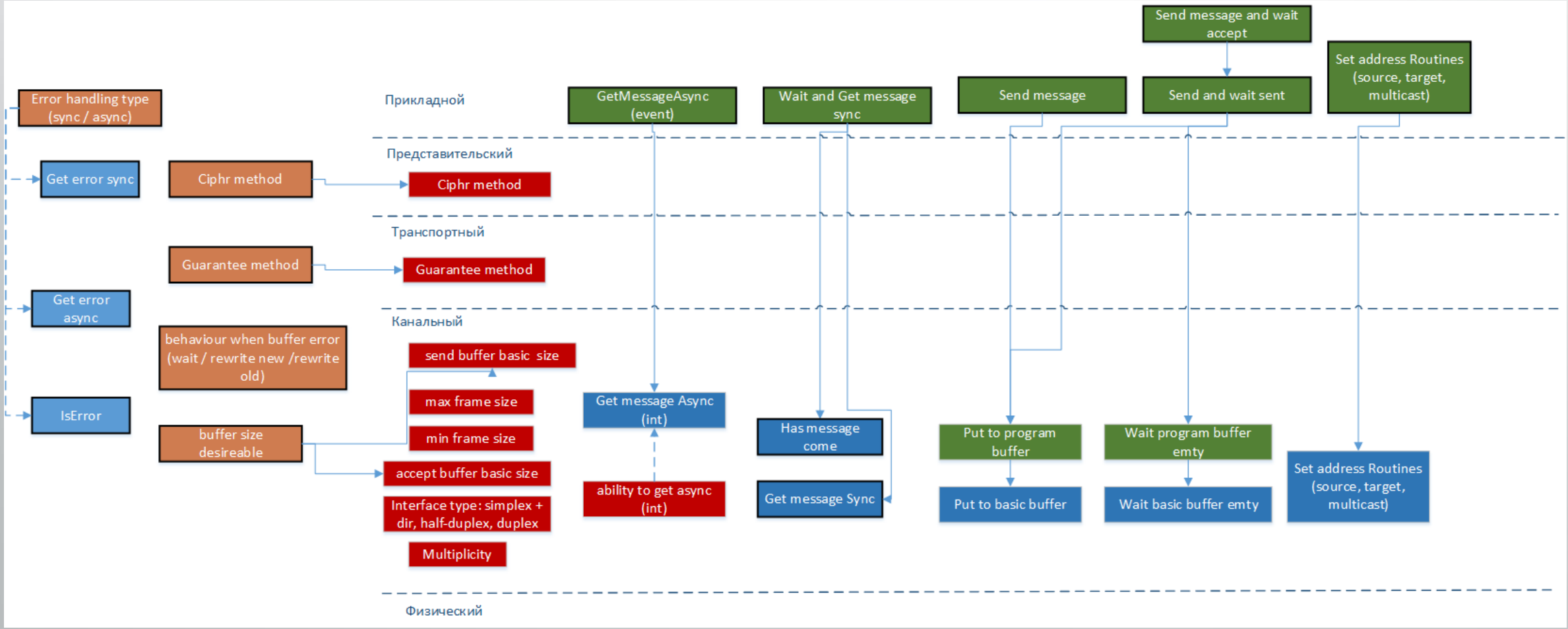
2) Сгенерировать код интерфейсного модуля, реализующий протокол с заданными разработчиком свойствами (нужна ли шифровка и какая, нужна ли гарантия ошибок и способ, способ уведомления об ошибках и т.д.).

3) Сгенерировать код отправляющего модуля по передаче сообщения и ожиданию ответа (если взаимодействие синхронное).

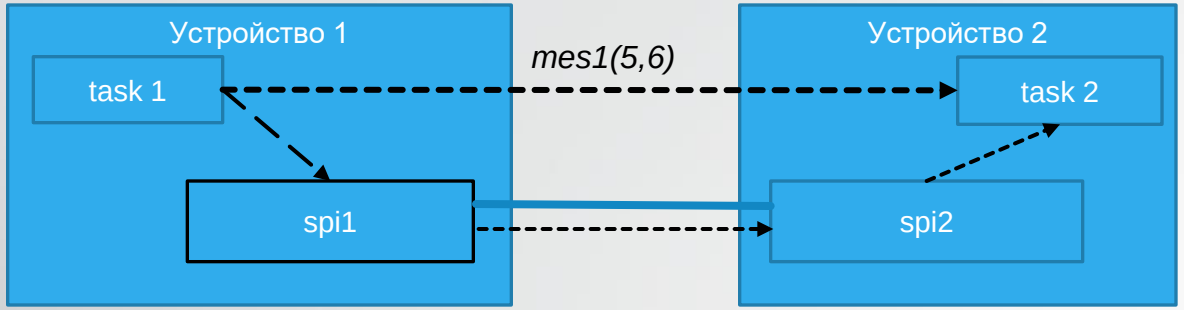
4) Сгенерировать код принимающего модуля по принятию сообщения и выдаче ответа о завершении принятия (при синхронном взаимодействии).

Унифицированная архитектура и API интерфейсных модулей

В результате анализа и обобщения наиболее распространенных интерфейсов (*1-Wire, SPI, UART RS-232 / Serial over USB, I2C, CAN, Bluetooth, Ethernet*) разработана унифицированная архитектура и API интерфейсных модулей. Показаны связи функций для генерации программного кода модулей.



Синтез алгоритма асинхронного взаимодействия между распределенными модулями



При синтезе асинхронного взаимодействия, сообщение посылается обработчику запросов интерфейсного модуля асинхронно, кладется в буфер, передается по интерфейсу, принимается интерфейсным модулем принимающего устройства и асинхронно отправляется в буфер целевой задаче.

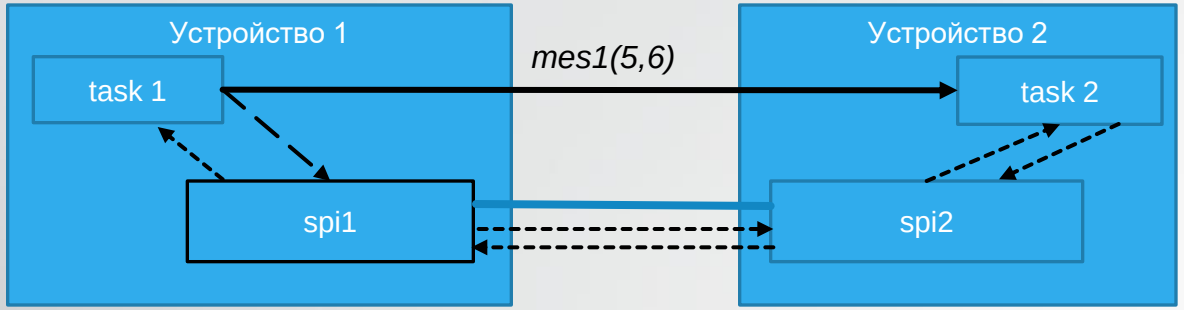
Служебная информация в пакете:

- ID адресата
- тип сообщения

	task1:	task2:
Embeddecy	send task2.mes1(5, 6) via spi1;	int x, y, a; accept mes1(x, y) do { a = x + y; … }

Генерируемый С	spi_reqhandler_sendAsync(task2_ID, task2_mes1, 5, 6);	int x, y, a;
		while (! task2_reghandler_hasmessage(task2_mes1);
		task2_reghandler_accept(task2_mes1, &x, &y);
		a = x + y;
		…

Синтез алгоритма синхронного взаимодействия между распределенными модулями



Служебная инф. в пакете (туда):
- ID адресата
- ID адресанта
- тип сообщения
- ID сообщения

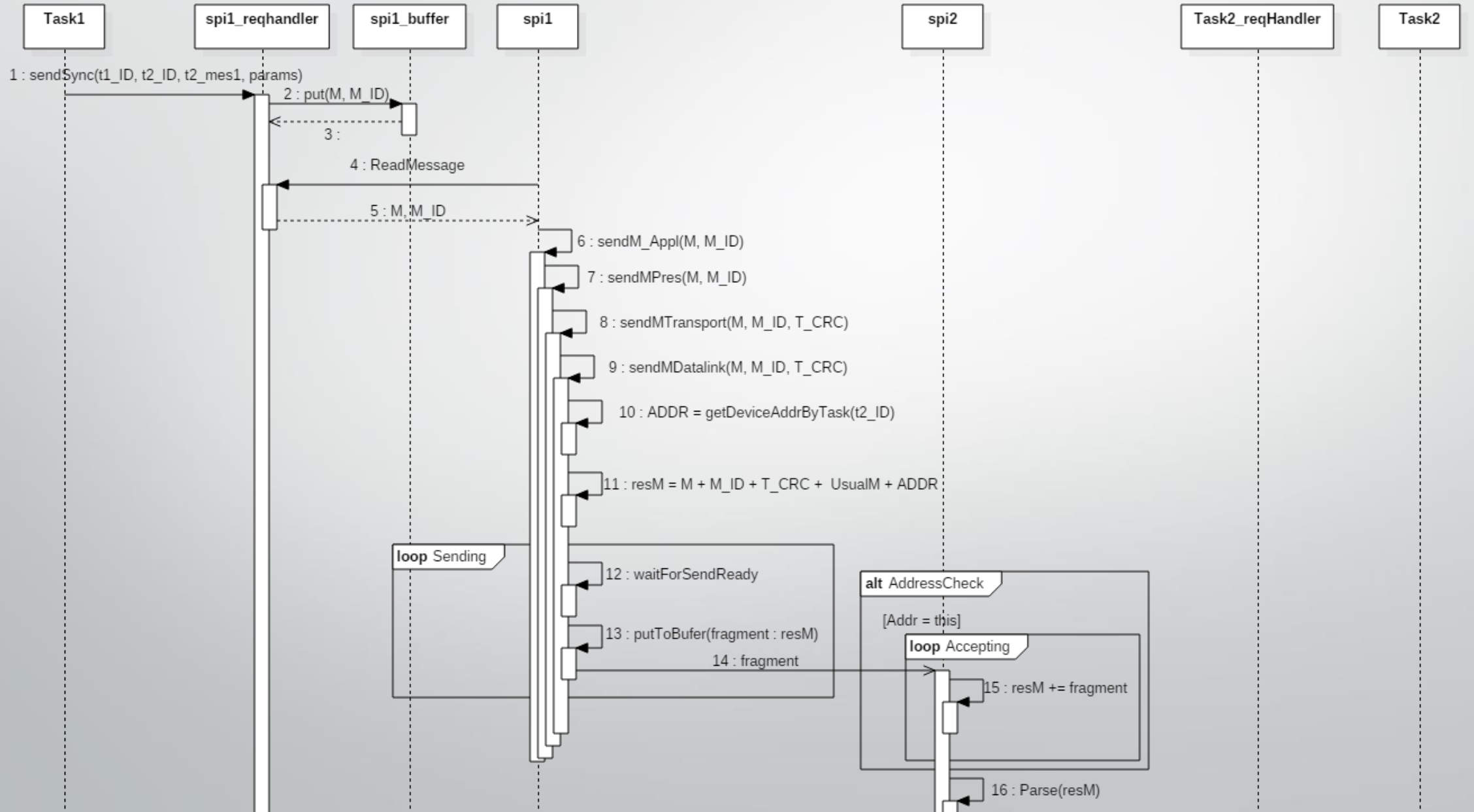
Служебная инф. пакета подтв.:
- ID адресанта
- ID сообщения

При синтезе синхронного взаимодействия, сообщение посылается обработчику запросов интерфейсного модуля асинхронно, кладется в буфер, передается по интерфейсу, принимается интерфейсным модулем принимающего устройства и асинхронно отправляется в буфер целевой задаче.

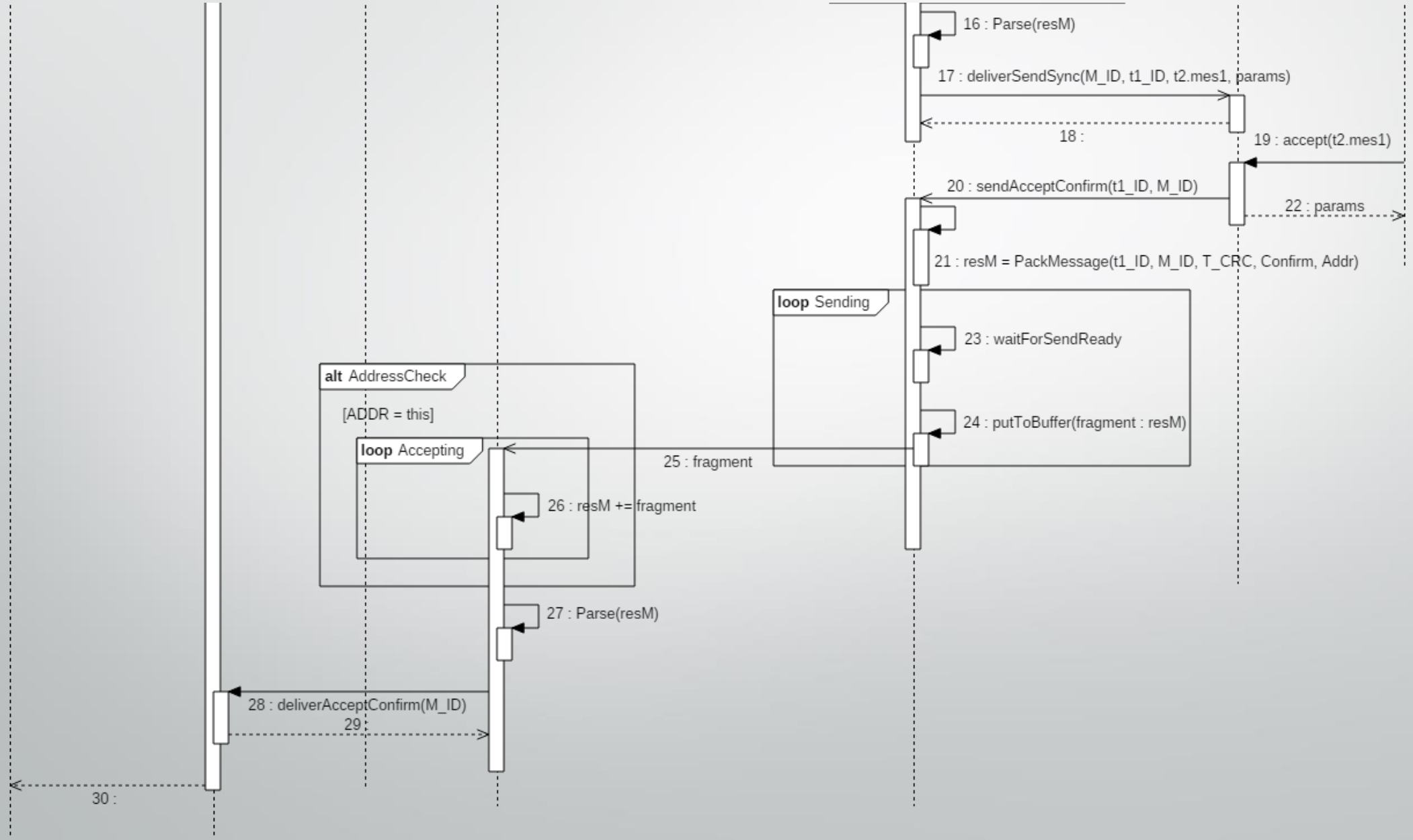
	task1:	**task2:**
Embeddecy	send task2.mes1(5, 6) via spi1;	int x, y, a; accept mes1(x, y) do { a = x + y; … }

Генерируемый С	spi_reqhandler_sendAsync(task2_ID, task2_mes1, 5, 6);	int x, y, a;
		while (! task2_reghandler_hasmessage(task2_mes1);
		task2_reghandler_accept(task2_mes1, &x, &y);
		a = x + y;
		…

Обобщенная схема алгоритма синхронного взаимодействия между распределенными модулями



Обобщенная схема алгоритма синхронного взаимодействия между распределенными модулями



Интерфейс обработчиков запросов задач

API обработчика запросов неинтерфейсной задачи

- void sendSync(m_type, m);
- void sendAsync(m_type, m);
- bool hasgot(m_type);
- accept(m_type);

API обработчика запросов интерфейсной задачи

- void sendSync(sender_t, target_T, m_type, m);
- void sendAsync(target_T, m_type, m);
- bool hasgot(m_type);
- accept(m_type);
- deliverSendSync(m_id, sender_t,m_type, m)
- sendAcceptConfirm(sender_t, m_id)
- deliverAcceptComfirm(m_id)

Параметры синтеза

Параметры синтеза программного кода по обеспечению **псевдопараллельного исполнения** задач:

- Тип многозадачности (выбор планировщика);
- Для кооперативной - допустимое количество низкоуровневых инструкций между инструкциями по передаче управления планировщику

Параметры синтеза программного кода по **реализации взаимодействия** модулей:

- Размер буфера неинтерфейсного модуля-задачи;
- Размер буфера интерфейсного модуля-задачи (сдвоенный: на принятие и отправку, заполнение с разных сторон);
- Максимальное одновременное количество сообщений, для которых ожидается подтверждение доставки (будет выделено соответствующее количество байт);
- Параметры протоколов:
  - Параметры интерфейса
  - Метод шифрования и его параметры
  - Метод доставки и его параметры
  - Поведение при ошибке буферизации: (ожидание освобождения, перезапись последнего, перезапись первого)

Генерация кода для конструкций, повышающих надежность языка

Элемент	Embeddecy	C

Пространство имен	`namespace a`
	`{`
	`  int b;`
	`  …`
	`}`
		Ко всем ID внутри пространства имен добавляется приставка пространства имен:
		
		`int a_b;`

Модули: пакеты и задачи	`package pack1`
	`{`
	`  public …`
	`  private …`
	`}`
		Генерируется два файла: .h + .c
		Public-элементы в .h-файле
		Private-элементы в .c файле

Удобная работа с битами	`PORTA[7] = 0;`	Обращения к битам реализуются через сдвиговые операции: `PORTA &= ~(1 << 7);`

Генерация кода для конструкций, реализующих механизм событий

Элемент	Embeddecy	C

Событие	`event myevent(int a, float b);`
	`myevent += … ;`
	`myevent -= … ;`
		1) При объявлении генерируется структура «событие»
		2) Функция добавления подписчика в событие `subscribe`
		3) Функция удаления подписчика `unsubscribe`

Анонимные функции	`myevent += (int a, float b) ->`
	`{`
	`  int k = a + b;`
	`};`
		Генерируется функция с произволильным именем:
		`void func123(int a, float b)`
		`{`
		`  int k = a + b;`
		`}`
		и подписывается на событие:
		`subscribe(myevent, func123);`

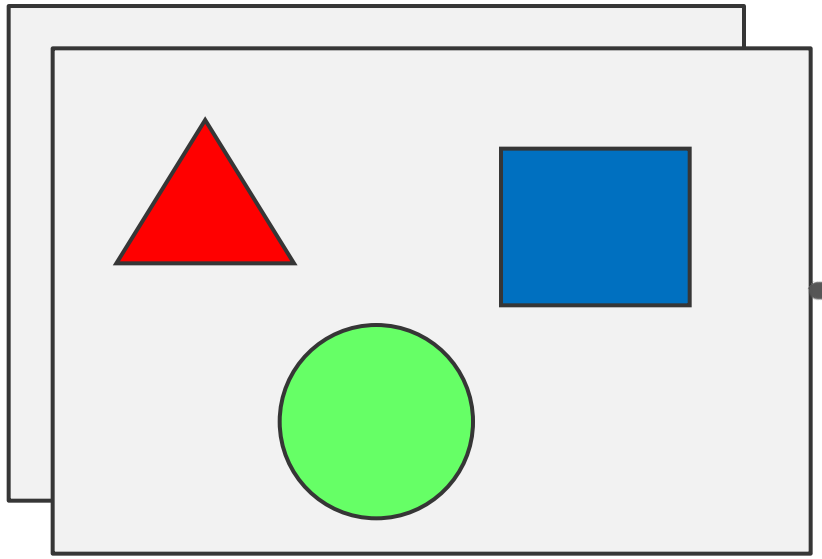
Делегат	`delegate void td(int, int);`	Генерируются две конструкции:
		1) объявляется типа ссылки на функцию:
		`typedef void (*) t_td(int, int);`
		2) объявляется переменная-ссылка:
		`t_td td;`

Стадии разработки МК-системы в рамках предложенного подхода

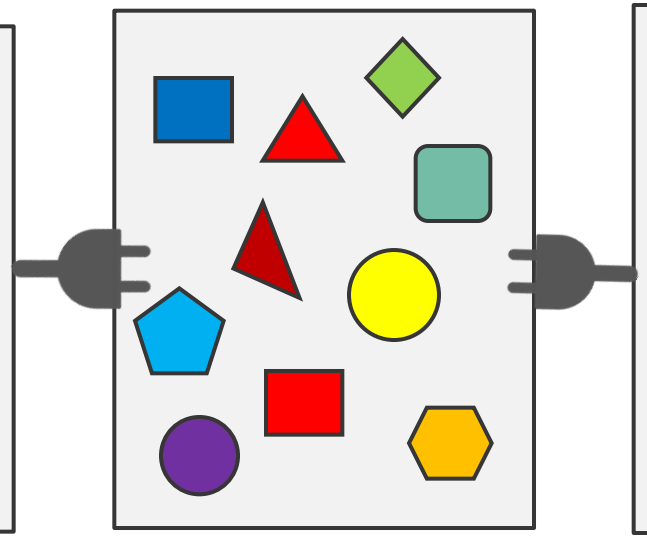
- Выбор и описание аппаратной конфигурации системы
- Декомпозиция задачи управления на функциональные модули с использованием библиотеки шаблонов
- Распределение функциональных модулей по исполнителям
- Проектирование межмодульного интерфейса – определение интерфейсных модулей
- Определение типа каждого из модулей:
  - Модуль-задача
  - Модуль-пакет
- Создание программных модулей на языке Embeddecy
- Генерация кода низкого уровня на языке С
- Компиляция кода низкого уровня на языке С стандартными средствами в исполняемый в гетерогенной среде код

Интегрированная среда разработки (IDE)

Графические редакторы



Библиотека шаблонов модулей



Редактор Embeddecy

```
#define_module spi1 = SPIControl<...>

task task2
{
   message mes1(int x);
   body
   {
         send task2.mes2() via spi1;
   }
}
```

Генерация кода на языке С

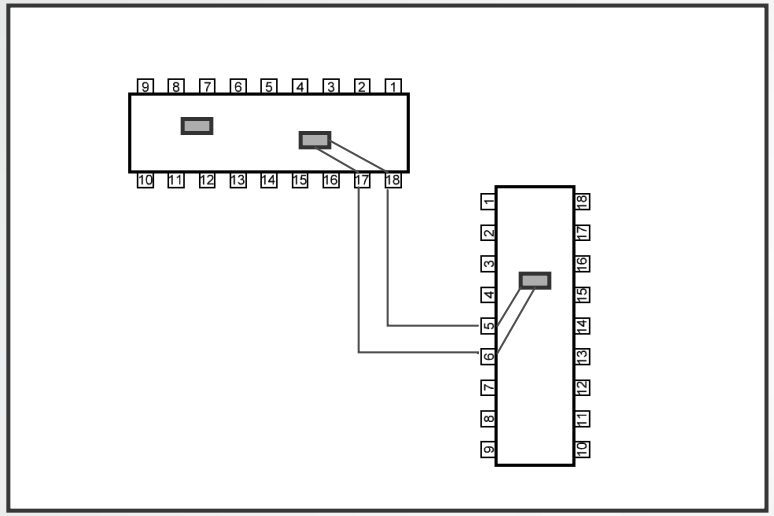
Программа на целевом ЯП (С)

Компиляция сторонними компиляторами

Бинарный образ

Связь редакторов системы

Редактор схемы оборудования



Генерация кода конфигурации шаблонов

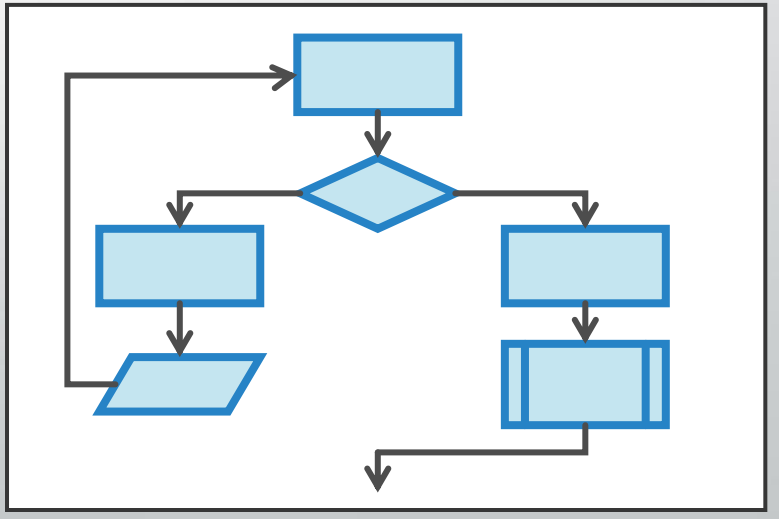
Генерация кода конфигурации шаблонов

Редактор Embeddecy

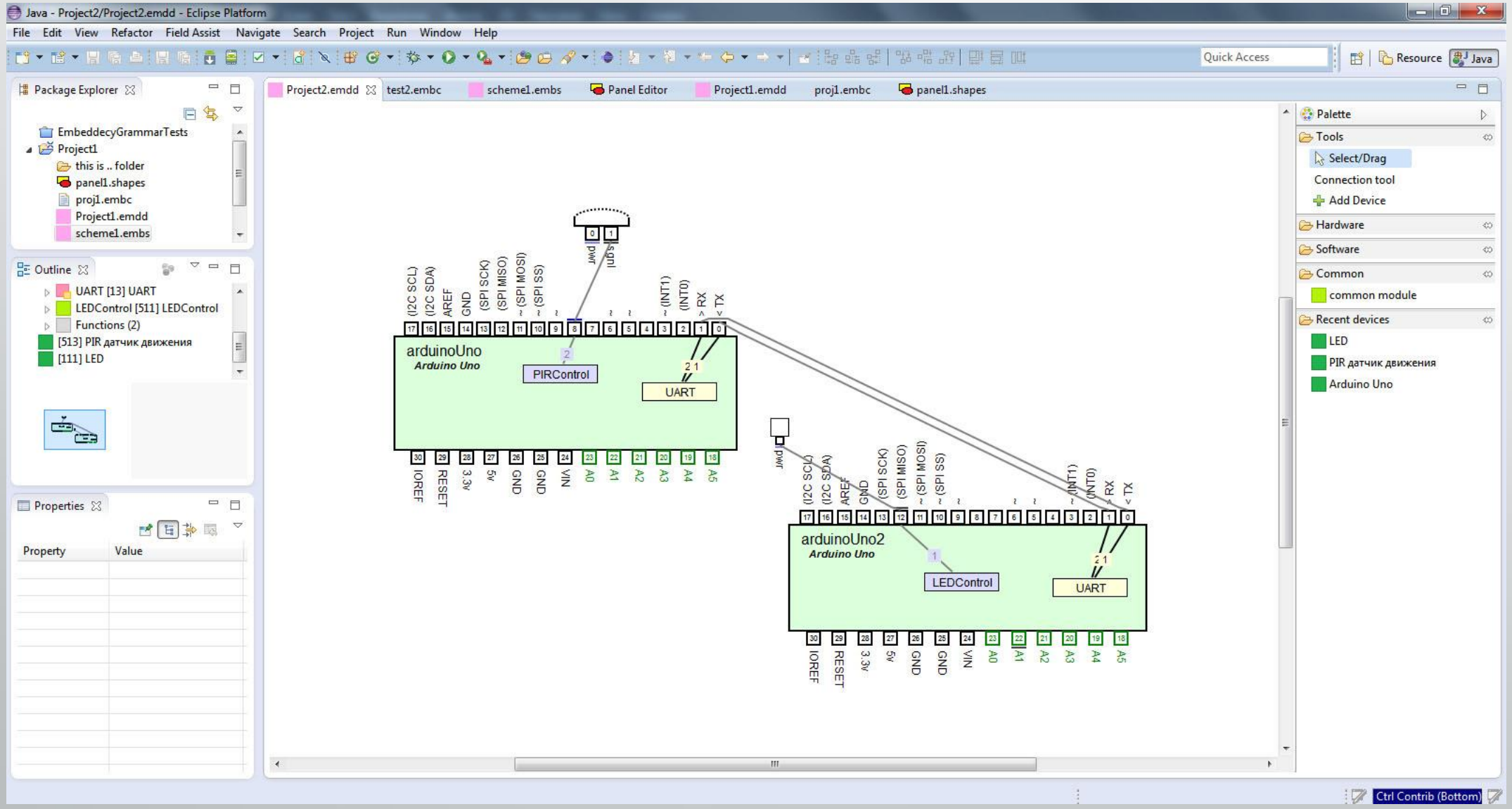
```
#define_module spi1 = SPIControl<...>

task task2
{
   message mes1(int x);
   body
   {
         send task2.mes2() via spi1;
   }
}
```

Формы представления программы

Редактор блок-схем



Конфигурация шаблонов в редакторе схемы оборудования



Редактор Embeddecy

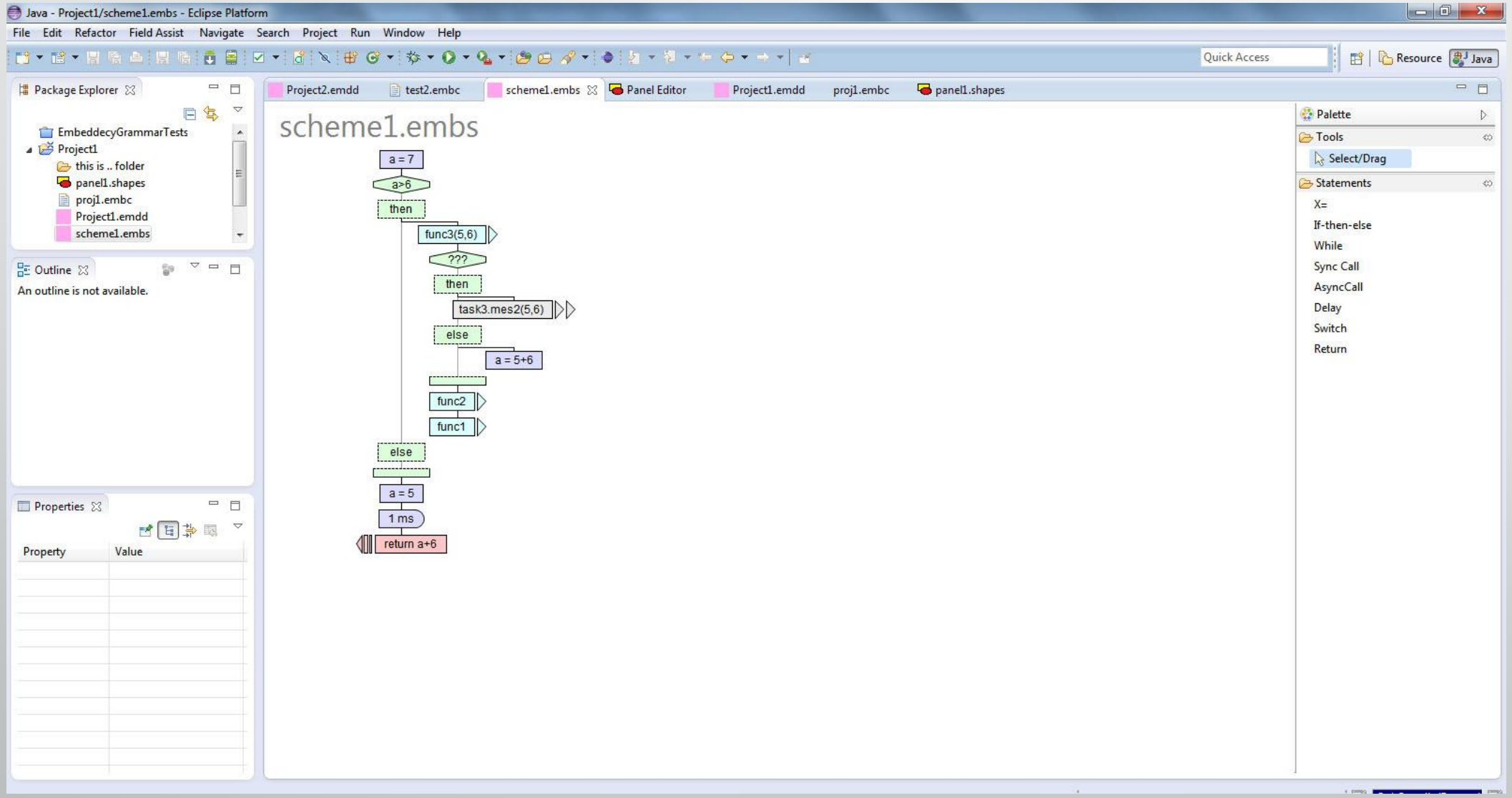
Код на Embeddecy для устройства 1

```
namespace arduinoUno
{
    #define_module pirc1 = PIRControl<pin pir_pin1>
    #define_module uart1 = UART<...>

    event event_onMove();
    event event_onNoMove();

    package pack1
    {
        #main
        void main()
        {
            pirc1.event_onMove += () ->
                {
                    arduinoUno2.led1.On() via uart1;
                };
            pirc1.event_onNoMove += () ->
                {
                    arduinoUno2.led1.Off() via uart1;
                };
        }
    }
}
```

Код на Embeddecy для устройства 2

```
namespace arduinoUno2
{
    #define_module led1 = LEDControl<pin led_pin>
    #define_module uart1 = UART<...>

    package pack1
    {
        #main
        void main()
        {
            while (true);
        }
    }

}
```

Прототип редактора блок-схем



Используемые технологии для реализации инструментов

Инструмент	Технология	Работы
IDE	Платформа Eclipse, язык Java	Разработана конфигурация плагинов, обеспечивающих работу компонентов в рамках общей IDE
Редактор Embeddecy	Платформа Xtext, antlr	Грамматика языка С переведена в нелеворекурсивную форму, разработана грамматика языка Embeddecy в нотации Xtext, разработаны модули автодополнения кода, подсветки синтаксиса
Графические редакторы схемы оборудования блок схем	Платформа GEF	Разработаны прототипы графических редакторов.
Генератор кода	Язык Xtend, EMF	Спроектирован генератор С-кода, реализован генератор некоторых конструкций программы Embeddecy, представленной в виде EMF модели AST дерева.

Научная новизна

- Модель представления распределенных программ для встраиваемых систем на базе микроконтроллеров
- Язык, поддерживающий разработку программ в рамках предложенной модели
- Алгоритмы синтеза программ управления для гетерогенных микроконтроллерных системы
  - Алгоритм генерации программного кода по обеспечению параллельного исполнения задач на микроконтроллере
  - Алгоритм генерации программного кода по обеспечению обмена сообщениями между параллельно работающими задачами

Содержание диссертации (1)

Содержание диссертации (2)

Публикации и доклады

ВАК:

1. Петров А.В., Большаков О. С., Лебедев А.С., Голубева Н.Е. Метод шаблонов приложений для повышения мобильности распределенных систем сбора и ретрансляции информации с биомедицинских датчиков // Журнал радиоэлектроники: электронный журнал. – 2013. – № 5.
2. Лебедев А.С., Большаков О. С., Петров А.В. Проектирование распределенной системы ретрансляции данных с мобильными клиентами на основе кроссплатформенных методов разработки программного обеспечения // Современные проблемы науки и образования. – 2013. – № 1.
3. Большаков О.С., Шаров В. Г., Петров А. В. Модель распределенных программ для встраиваемых систем // Вестник Рыбинского государственного авиационного технического университета имени П. А. Соловьева. – Рыбинск, 2015. – №1(32). – С. 165-171.
4. Большаков О.С., Шаров В. Г., Петров А. В. Реализация кроссплатформенности программ для специализированных распределенных встраиваемых систем // Вестник Череповецкого государственного университета. – Череповец, 2015. – №7(68). - С. 8-13.

Прочие:

3. Большаков О. С. Петров А. В. Разработка программной системы автоматизации аппаратного комплекса лабораторных стендов // Компьютерная Интеграция Производства и ИПИ-технологии: Сборник трудов IV всероссийской научно-практической конференции. – Оренбург: ГОУ ОГУ, 2009. – С. 409-414
4. Большаков О. С. Петров А. В. Верификация управляющих программ на примере системы автоматизации робототехнического стенда // Теория и практика системного анализа: Труды I Всероссийской научной конференции молодых ученых. – Т. II. – Рыбинск: РГАТУ имени П. А. Соловьева, 2010. – С. 109 – 116.
5. Большаков О. С. Петров А. В. Модель представления программ управления распределенными микроконтроллерными системами // Теория и практика системного анализа: Труды III Всероссийской научной конференции молодых ученых. – Т. I. – Рыбинск: РГАТУ имени П. А. Соловьева, 2014. – С. 96 – 107.
6. Большаков О. С., Шаров В. Г., Петров А. В. Модель распределенных программ для встраиваемых систем // Актуальные проблемы технических наук : сборник статей Международной научно-практической конференции. – Уфа: АЭТЕРНА, 2015. – С. 30-34.

Патенты

1. Патент №115526 на **полезную модель** “Устройство формальной проверки согласованности управляющей программы микроконтроллерной системы заданной спецификации”
2. Свидетельство о государственной регистрации **программы для ЭВМ** № 2013613819 “Интегрированная среда разработки управляющих программ для робототехнических стендов РГАТУ”
3. Свидетельство о государственной регистрации **программы для ЭВМ** №2013616323 “Интегрированная среда разработки программ для микроконтроллеров AVR”
4. Патент на **изобретение** № 2553067 “Способ предоставления информационной поддержки разработчика программного обеспечения для микроконтроллеров и реализующая его система”

Качество программного обеспечения

Модель качества программного обеспечения (ISO 9126-1)

Предложенные средства описания и автоматического синтеза программ позволяют повысить эффективность разработки и качество программного обеспечения за счет:

- повышения уровня программирования,
- улучшения степени структуризации программного кода, в том числе с точки зрения параллельного исполнения модулей,
- снижения вероятности допущения ошибок при реализации низкоуровневых алгоритмов взаимодействия, синтезируемых в предлагаемой системе автоматически.
- улучшения контроля за степенью портируемости программы за счет выделения кроссплатформенных элементов интерфейса и аппаратно-зависимых.
- использования языка программирования, поддерживающего предложенную высокоуровневую модель представления программ и содержащего современные средства синтаксической выразительности

Доп. слайд

Стадии проектирования встраиваемой системы на базе МК

Доп. слайд

Модель представления программ

Особенности модели:

- Модель расширяет существующую модель представления программ в рамках структурного программирования, т.е. имеет с ней совместимость, добавляя новые уровни абстракций.
- Изменения в шаблонах модулей: добавлен параметр-модуль, возможно задать интерфейс модуля и ограничения на параметр-тип. Это позволяет создать графический конфигуратор модулей управления.
- Модель содержит особый вид модулей-интерфейсные, что является результатом обобщения функций интерфейсов микроконтроллеров и позволяет генерировать реализацию межустройстного взаимодействия
- Модель позволяет описывать параллельные программы и предоставляет средства синхронизации между параллельным задачами
- В отличие от модели, лежащей в основе языка Ada, к возможности синхронного взаимодействия задач (механизм рандеву) добавлена возможность асинхронного взаимодействия.
- Добавлена концепция событий, что позволяет задачам, в отличие от модели Ada, быть инициаторами событий для подписчиков.