Знакомство с новыми микроконтроллерами SMART ARM компании Atmel: работаем с модулем CCL микроконтроллера SAM L21

Николай АРТЕМОВ
anv@efo.ru

Современные тенденции, направленные на миниатюризацию встраиваемых решений, побуждают производителей электронных компонентов интегрировать в микроконтроллеры все большее количество различных узлов в качестве периферийных модулей. Компания Atmel при разработке нового семейства микроконтроллеров SAM L21 постаралась получить продукт с обширным и разнообразным набором периферийных модулей, на базе которого можно спроектировать встраиваемое решение для большинства инженерных задач. Данная статья посвящена работе с модулем программируемой логики CCL микроконтроллера Atmel SAM L21. В ней мы рассмотрим функциональные возможности модуля и базовые принципы работы с ним.

Во многих проектах, с точки зрения эффективного распределения ресурсов, выгодно часть операций перенести с микроконтроллера на внешние логические элементы. Это обусловлено разными причинами: требованием высокой производительности, желанием обеспечить автономность работы узла или загруженностью вычислительного ядра. Но использование дополнительных элементов идет вразрез с современными тенденциями, направленными на миниатюризацию устройств. И в такие моменты разработчику приходится выбирать, чем жертвовать. Одним из немногих бескомпромиссных решений является интеграция простых логических элементов в микроконтроллер. Подобное решение редко встречается в настоящее время, тем более

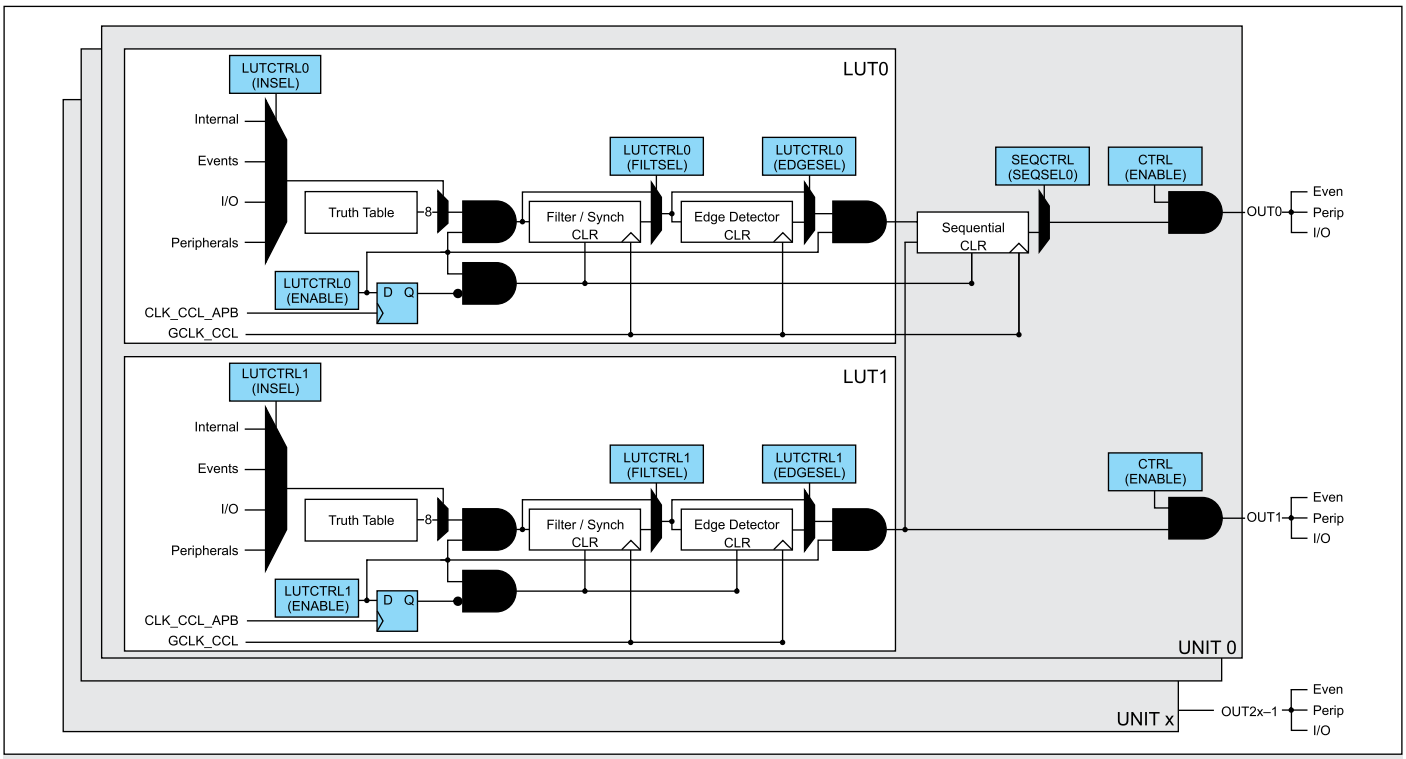

Рис. 1. Функциональная схема периферийного модуля программируемой логики CCL

в младших семействах на базе архитектуры ARM, где это больше всего востребовано.

Микроконтроллеры нового семейства Atmel SAM L21, вобравшие передовые достижения микроэлектроники, имеют в своем обширном наборе периферии и модуль программируемой логики CCL (Configurable Custom Logic). CCL — периферийный модуль программируемой логики, позволяющий реализовать:

- логические операции AND, NAND, OR, NOR, XOR, XNOR, NOT;
- составные логические операции;
- последовательные логические операции: D Latch (по уровню), D flip-flop (по фронту), JK-, RS-триггеры.

Функциональная схема модуля CCL представлена на рис. 1. Он включает четыре модуля реализации логических функций (LUT) и два узла реализации последовательных логических функций (Sequential). Каждый модуль LUT содержит:

- входной коммутатор;
- таблицу истинности (**Truth Table**);
- отключаемый узел фильтрации и синхронизации (**Filter/Synch**);
- отключаемый детектор фронта (**Edge Detector**).

Входной коммутатор позволяет подключить вход LUT к линиям GPIO, шине Events или периферийным устройствам — аналоговому компаратору (AC), таймеру/счетчику (TC), интерфейсному модулю SERCOM. Предусмотрен также режим FEEDBACK, в котором выход LUT замкнут на свой вход. Каждый вход может быть настроен как прямой или инверсный.

Таблица истинности (**Truth Table**), представленная на рис. 2, позволяет реализовать любые логические операции, описав их. Для описания логической функции необходимо ввести параметр, представляющий собой 8-битное число, в котором каждый бит описывает значение логической функции для входной комбинации из трех операндов. Значения операндов являются адресом, указывающим место бита в числе. Пример описания логических операций в таблице истинности приведен в таблице 1.

Отключаемый узел фильтрации и синхронизации (**Filter/Synch**) предназначен для сглаживания выбросов и помех, а также для синхронизации сигналов, поступающих

Таблица 1. Пример описания логических функций в таблице истинности

IN[]	TRUTH	AND	NAND	OR	NOR	XOR	XNOR	NOT
000	TRUTH[0]	0	1	0	1	0	1	1
001	TRUTH[1]	0	1	1	0	1	0	X
010	TRUTH[2]	0	1	1	0	1	0	X
011	TRUTH[3]	0	1	1	0	0	1	X
100	TRUTH[4]	0	1	1	0	1	0	X
101	TRUTH[5]	0	1	1	0	0	1	X
110	TRUTH[6]	0	1	1	0	0	1	X
111	TRUTH[7]	1	0	1	0	1	0	0
	TRUTH value	0×80	0×7F	0×FE	0×01	0×96	0×69	0×01

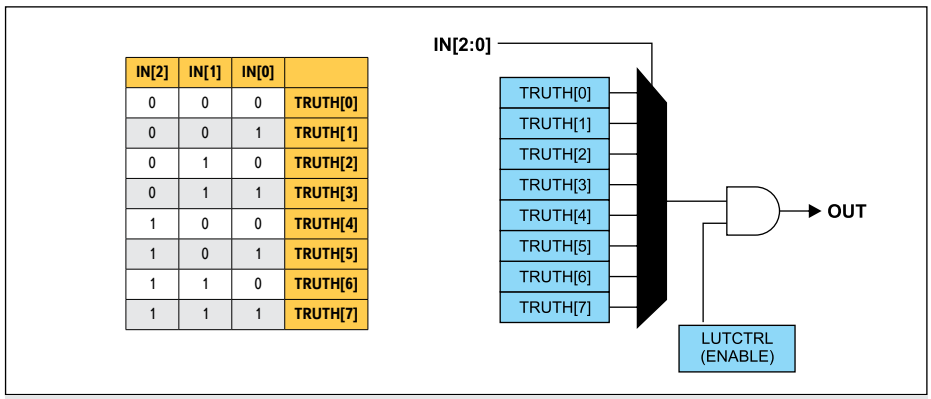

Рис. 2. Таблица истинности периферийного модуля программируемой логики

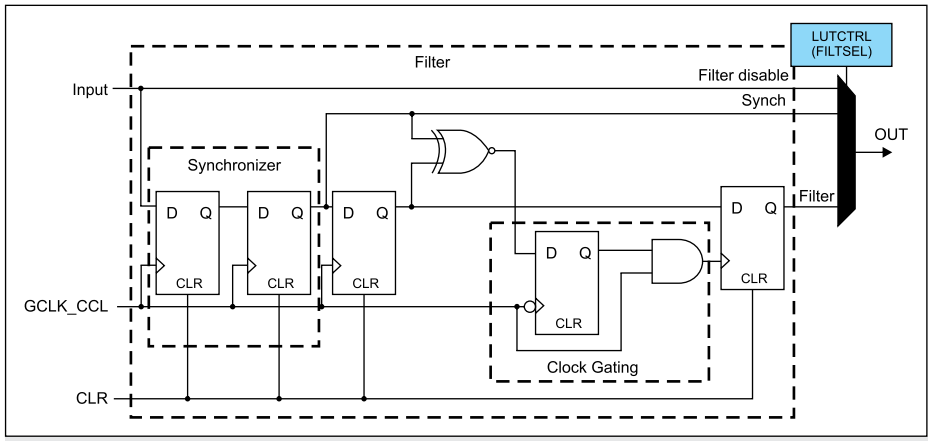

Рис. 3. Функциональная схема узла фильтрации/синхронизации модуля LUT

на вход LUT. Функциональная схема данного узла представлена на рис. 3. Стоит обратить внимание на то, что включенная синхронизация вносит задержку сигнала на 2 такта, синхронизация с фильтрацией — на 4 такта.

Узел детектирования фронта (**Edge Detector**) позволяет выделить фронт сигнала. Данный узел, как и узел фильтрации и синхронизации, может быть отключен.

Узел последовательной логики (**Sequential**) позволяет реализовывать распространенные триггеры: D Latch, D flip-flop, JK, RS. Здесь источником сигнала являются два модуля LUT, каждый из которых влияет на работу. Например, при реализации RS-триггера при помощи узла Sequential выход LUT0 или LUT2 будет подключен к входу SET триггера, а выходы LUT1 или LUT3 к входу RESET (рис. 4).

Модуль CCL допускает последовательное соединение модулей LUT для реализации сложных функций, но есть ограничение: каждый выход модуля LUT может быть подключен к входу только предыдущего модуля (рис. 5.)

Выходы модуля программируемой логики CCL могут быть подключены к выводам микроконтроллера или к шине Event System для формирования событий для других периферийных модулей.

Для оценки возможностей модуля CCL микроконтроллера Atmel SAM L21 воспользуемся отладочной платой SAM L21-XPRO (рис. 6) с дополнительным модулем расширения OLED1 Xplained Pro (рис. 7). Для написания и компиляции программного кода применим интегрированную среду разработки IDE Atmel Studio 7. Atmel Studio 7 в сочетании с обширным набором библиотек из рабочего окружения разработчика Atmel Software Framework (ASF) значительно упрощает и ускоряет процесс создания и отладки приложений для микроконтроллеров Atmel.

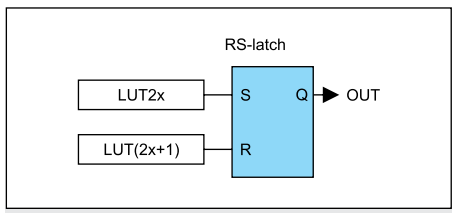

Рис. 4. Узел Sequential в режиме RS-триггера

Для знакомства с модулем CCL создадим простую тестовую программу, которая сконфигурирует модуль CCL следующим образом:

1. Соединит модули LUT последовательно LUT0–LUT3–LUT2–LUT1.
2. Входы LUT0 подключит к внешним выводам микроконтроллера PB10, PB11, PA15, на которые заведены клавиши BUTTON 1, BUTTON 2, BUTTON 3 модуля расширения OLED1 Xplained Pro.

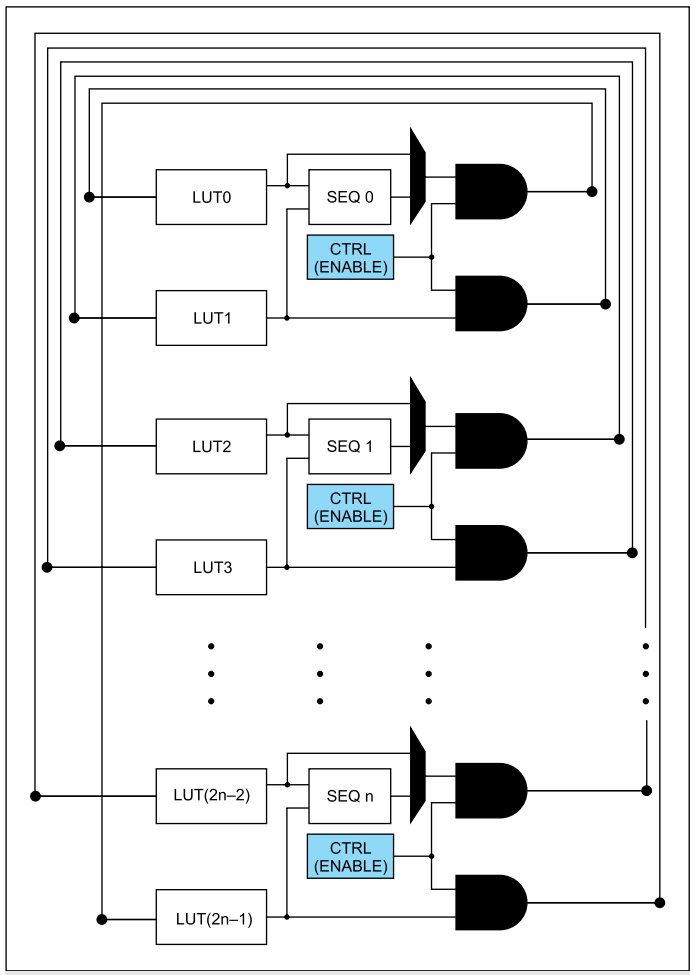

Рис. 5. Последовательное соединение модулей LUT

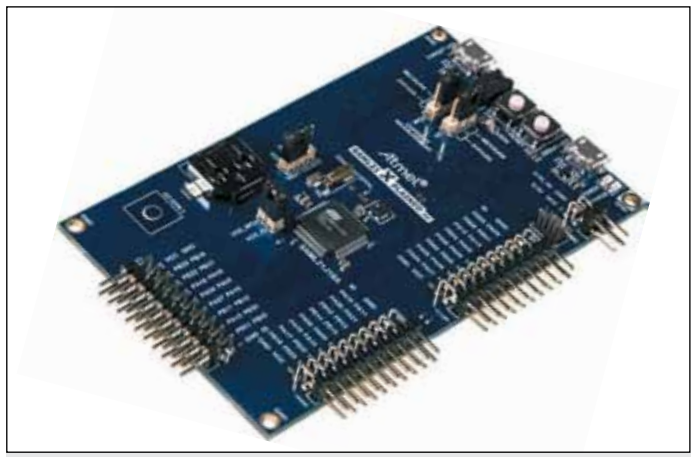

Рис. 6. Отладочная плата SAM L21-XPRO

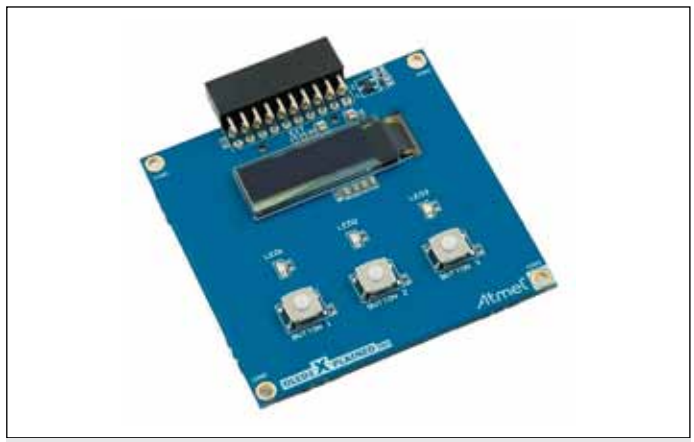

Рис. 7. Модуль расширения OLED1 Xplained Pro

3. Выход LUT1 подключит к внешнему выводу микроконтроллера PB00, к которому подключен светодиод LED2 модуля расширения OLED1 Xplained Pro.

Модуль CCL настроим так, чтобы только при одновременном нажатии клавиш **BUTTON 1** и **BUTTON 3** включался светодиод LED2. После настройки модуля CCL микроконтроллер переходит в режим standby. Наша тестовая программа реализует популярную задачу декодирования комбинации клавиш разблокировки клавиатуры или активации устройства без участия процессорного ядра. Конфигурация модуля CCL, формируемая тестовой программой, представлена на рис. 8.

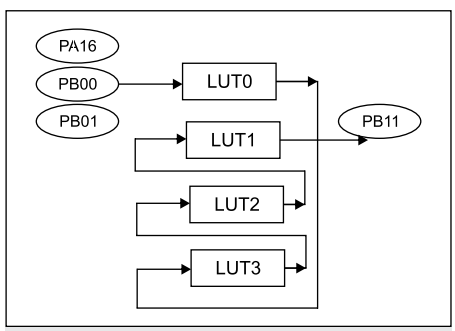

Рис. 8. Конфигурация модуля CCL, формируемая тестовой программой

При реализации программы существует ряд ограничений:

- все клавиши модуля расширения OLED1 Xplained Pro подтянуты к шине питания VCC, уровень сигнала отжатой клавиши соответствует логической «1», нажатой — «0»;
- светодиод LED1 модуля расширения OLED1 Xplained Pro включается уровнем, соответствующим логическому «0».

Исходя из этих условий, сформируем таблицу истинности для LUT0. Данные для Truth Table представлены в таблице 2. Для всех комбинаций входных данных, кроме «010», указываем значение TRUTH, равное «1» (светодиод выключен), а для комбинации «010»–«0» (светодиод включен). Переводим значение в столбце TRUTH в шестнадцатеричный код, получаем 0xFB — это и есть значение для Truth Table модуля LUT0.

Таблица 2. Таблица истинности LUT0

BUTTON 1 (IN[0])	BUTTON 2 (IN[1])	BUTTON 3 (IN[2])	TRUTH
0	0	0	1
0	0	1	1
0	**1**	**0**	**0**
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1
			0xFB

Все исходные данные для написания тестовой программы готовы. Запускаем с Atmel Studio 7, создаем новый проект GCC C ASF Board Project (**File/New/Project**), указав имя нашего проекта в поле Name. Выбираем интересующий нас микроконтроллер ATSAML21J18A из списка, установив флаг Select by Device. Проверяем, что в выпадающем списке Extensions выбрана наиболее свежая версия ASF 3.27 или старше. Указываем необходимую целевую плату SAM L21 Xplained Pro (ATSAML21J18A)-ATSAML21J18A, предварительно установив флаг Select by Board. Выбрав микроконтроллер, отладочную плату и нажав клавишу **OK**, заканчиваем создание проекта.

После создания проекта отобразится окно **ASF Wizard**, где мы можем подключить необходимые нам компоненты из окружения ASF. Для этого нужно выбрать наш проект в выпадающем списке Project, после чего программа автоматически подгрузит все доступные для данного микроконтроллера компоненты (рис. 9). Загружаем в наш про-

ект драйвер для работы с модулем программируемой логики CCL-Configurable Custom Logic (driver). Для этого выбираем необходимый компонент, нажимаем клавишу **Add**, затем — **Apply**.

Теперь можем приступать к написанию самой программы. Открываем файл *main.c*, создаем три структуры для хранения настроек CCL, LUT и конфигурации портов ввода/вывода:

```
struct ccl_config ccl_conf;
struct ccl_lut_config ccl_lut_conf;
struct system_pinmux_config pinmux_conf;
```

Инициализируем систему, считываем настройки портов ввода/вывода по умолчанию, настраиваем порт PA16 как вход CCL, включаем подтягивающий резистор:

```
system_init();
system_pinmux_get_config_defaults( &pinmux_conf );
pinmux_conf.mux_position = MUX_PA16I_CCL_IN0;
pinmux_conf.input_pull = true;
```

Записываем настройки в регистры микроконтроллера:

```
system_pinmux_group_set_config( &PORTA, PORT_PA16I_CCL_
IN0, &pinmux_conf );
```

Настраиваем порты PB00, PB01 как входы CCL, а порт PB11 настраиваем как выход CCL:

```
system_pinmux_group_set_config( &PORTB,PORT_PB00I_CCL_
IN1| PORT_PB01I_CCL_IN2
                | PORT_PB11I_CCL_OUT1, &pinmux_conf );
```

Конфигурируем модуль CCL. Считываем настройки CCL по умолчанию, настраиваем источник тактирования, активируем работу в режиме standby, инициализируем CCL с нашими настройками:

```
ccl_get_config_defaults( &ccl_conf );
ccl_conf.clock_source = GCLK_GENERATOR_0;
ccl_conf.run_in_standby = true;
ccl_init(&ccl_conf);
```

Переходим к настройкам модулей LUT. Считываем настройки модуля в структуру ccl_lut_conf, загружаем в таблицу истинности значение из таблицы 2, равное 0xFB, которое мы рассчитали ранее:

```
ccl_lut_get_config_defaults( &ccl_lut_conf );
ccl_lut_conf.truth_table_value = 0xFB;
```

Подключаем входы LUT0 к выводам микроконтроллера, записываем новые параметры в регистры LUT0:

```
ccl_lut_conf.input0_src_sel = CCL_LUT_INPUT_SRC_IO;
ccl_lut_conf.input1_src_sel = CCL_LUT_INPUT_SRC_IO;
ccl_lut_conf.input2_src_sel = CCL_LUT_INPUT_SRC_IO;
ccl_lut_set_config( CCL_LUT_0, &ccl_lut_conf );
```

Конфигурируем LUT1. Как и в предыдущем случае, считываем параметры по умолчанию, загружаем в таблицу истинности значение 0x02 (LUT1 будет работать как повторитель входного сигнала):

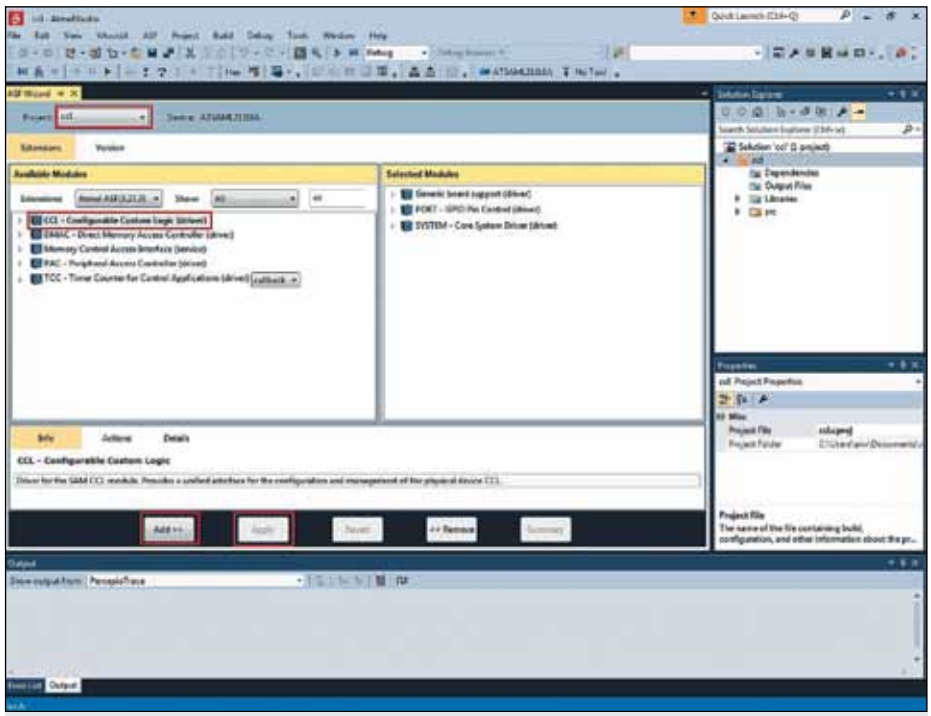

Рис. 9. Загрузка программных компонентов ASF

```
ccl_lut_get_config_defaults( &ccl_lut_conf );
ccl_lut_conf.truth_table_value = 0x02;
```

Вход LUT1 подключаем к выходу LUT2, записываем параметры в регистры микроконтроллера:

```
ccl_lut_conf.input0_src_sel = CCL_LUT_INPUT_SRC_LINK;
ccl_lut_set_config( CCL_LUT_1, &ccl_lut_conf );
```

Аналогично настраиваем модули LUT2 и LUT3:

```
// LUT2
ccl_lut_get_config_defaults( &ccl_lut_conf );
ccl_lut_conf.truth_table_value = 0x02;
ccl_lut_conf.input0_src_sel = CCL_LUT_INPUT_SRC_LINK;
ccl_lut_set_config( CCL_LUT_2, &ccl_lut_conf );
// LUT3
ccl_lut_get_config_defaults( &ccl_lut_conf );
ccl_lut_conf.truth_table_value = 0x02;
ccl_lut_conf.input0_src_sel = CCL_LUT_INPUT_SRC_LINK;
ccl_lut_set_config( CCL_LUT_3, &ccl_lut_conf );
```

Модули LUT готовы к работе, активируем их, а также сам модуль CCL:

```
ccl_lut_enable(CCL_LUT_0);
ccl_lut_enable(CCL_LUT_1);
ccl_lut_enable(CCL_LUT_2);
ccl_lut_enable(CCL_LUT_3);
ccl_module_enable();
```

Теперь переводим микроконтроллер в режим Standby:

```
system_set_sleepmode(SYSTEM_SLEEPMODE_STANDBY);
system_sleep();
```

Программа готова, можем загружать ее в микроконтроллер и проверять ее работоспособность.

На примере этой простой программы мы увидели, как без использования ядра микроконтроллера и при помощи модуля программируемой логики можно выполнять обработку сигнала. Благодаря широким возможностям по настройке модуля CCL область решаемых задач намного шире простых логических операций. С помощью CCL можно:

- преобразовывать последовательные посылки в параллельные;
- формировать выходной управляющий сигнал или выполнять последующую обработку выходного сигнала аналогового компаратора;

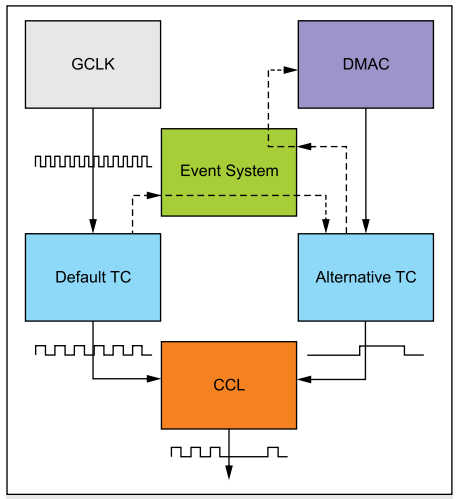

Рис. 10. Функциональная схема алгоритма кодирования данных для передачи по ИК-интерфейсу

- расширять функциональные возможности таймера/счетчика по захвату событий;
- создавать дополнительные функции для интерфейсного модуля SERCOM;
- выполнять модулирование сигнала;
- реализовывать делитель частоты входного сигнала.

Пример реализации алгоритма кодирования данных для передачи по ИК-интерфейсу, представленный на рис. 10, наглядно демонстрирует возможности совместной работы CCL с другими периферийными модулями.

Возможность программного многократного переконфигурирования модуля CCL при настройке и работе устройства делает подобное решение более адаптивным в сравнении с классическим решением на внешних стандартных логических элементах.

Данный пример демонстрирует широкие возможности нового микроконтроллера SAM L21. Использование библиотеки готовых компонентов Atmel Software Framework (ASF) и интегрированной среды Atmel Studio 7 значительно упрощает процесс программирования модуля CCL. В следующих статьях мы продолжим знакомить читателей с особенностями работы периферийных блоков SAM L21.