SMART ARM-based Microcontrollers

AT03716: Implementation of SAM L Configurable Custom Logic (CCL) Peripheral

APPLICATION NOTE

Description

The Configurable Custom Logic (CCL) module contains programmable logic for creating logic gates, sequential logic or a combination of the both. Inputs can either be I/O pins, internal feedback, peripherals, or events from the Event System. With CCL, logic operations can be performed without CPU intervention. This application note presents various features of the CCL, along with an example application.

The implementation shown in this application note is applicable for the following devices:

- Atmel®| SMART SAM L21
- Atmel | SMART SAM L22

Table of Contents

Description....1

1. Module Overview....3
   - 1.1. Truth Table Inputs....3
     - 1.1.1. Masked Input (MASK)....4
     - 1.1.2. I/O Pin Input (IO)....4
     - 1.1.3. Linked LUT Input (LINK)....4
     - 1.1.4. Internal Feedback Input (FEEDBACK)....4
     - 1.1.5. Internal Events Input (EVENT)....4
     - 1.1.6. Analog Comparator Input (AC)....5
     - 1.1.7. Timer/Counter Inputs (TC and ALTTC)....5
     - 1.1.8. Timer/Counter for Control Application Inputs (TCC)....5
     - 1.1.9. Serial Communication Output Transmit Inputs (SERCOM)....6
   - 1.2. Programmable Lookup Table....6
   - 1.3. Filter and Edge Detector....7
   - 1.4. Sequential Logic....7

2. Example Overview....10
   - 2.1. Prerequisites....11
   - 2.2. Peripherals....11
   - 2.3. Configurations....12
     - 2.3.1. Configure CCL....12
     - 2.3.2. Configure TC....12
     - 2.3.3. Configure DMAC....13
     - 2.3.4. Configure EVSYS....13
     - 2.3.5. Edit Configuration Files....13
   - 2.4. Main Routine....13

3. Software License....14

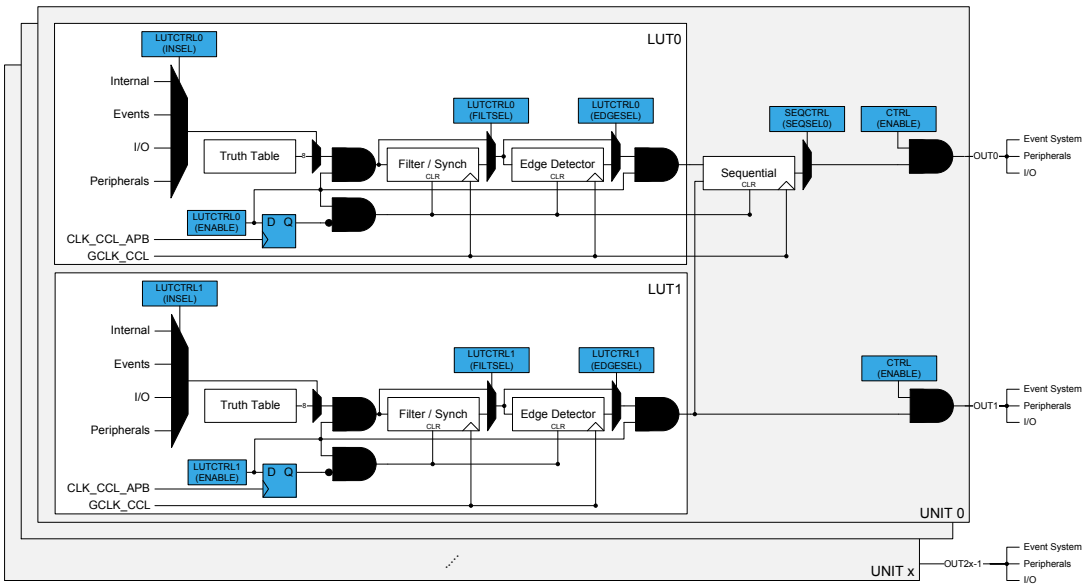
4. Revision History....15

1. Module Overview

The Configurable Custom Logic (CCL) module contains programmable logic which can be connected to I/O pins, internal feedback, peripherals or events. This allows the user to eliminate logic gates on the PCB for glue logic functions, and introduces additional functionality to applications. CCL can for instance be used to implement conditional wakeup, where a specific combination of pin and peripheral states enables the device to wakeup, or optionally to enter a safe state if necessary.

A key element of CCL is the Programmable Lookup Table (LUT). Each LUT consists of three individually configurable inputs and a truth table, in addition to optional filter, synchronizer and edge detector. Each LUT can generate an output as a user programmable logic expression from the three inputs. As can be seen from following block diagram, two LUTs (LUT0 and LUT1, LUT2 and LUT3) are connected to a sequential logic block, forming a unit. The optional sequential logic can be enabled for complex waveform generation such as JK flip-flop, RS latch, D latch or gated D flip-flop. The output can be routed to I/O pins, peripherals, event system or as feedback to the corresponding unit. The output can also be applied as feedback to a different LUT.

Figure 1-1. Block Diagram



1.1. Truth Table Inputs

The inputs IN[x] to the LUT can be selected from the following sources:

- Masked
- I/O Pin
- Linked LUT
- Internal Feedback
- Events
- Peripherals:
  - Analog Comparator output (AC)
  - Timer/Counters waveform outputs (TC/TCC)
  - Serial Communication output transmit interface (SERCOM)

1.1.1. Masked Input (MASK)

This is the default option for the inputs. When an input is masked, the value on the corresponding input will always be zero. If only two inputs are of interest, the unused input should be masked to ensure that its value is constant. Masked input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x0.

1.1.2. I/O Pin Input (IO)

I/O pins can be applied as inputs for the LUTs. An overview of available LUT I/O pins can be found in the "PORT Function Multiplexing" table in the "I/O Multiplexing and Considerations" chapter in the device datasheet. I/O pin input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x4.

To enable a peripheral function on a pin, the Peripheral Multiplexer Enable bit in the Pin Configuration register (PINCFGn.PMUXEN) corresponding to specific pin must be written to one. The selection of peripheral function is done by writing to the Peripheral Multiplexing Odd and Even bits in the Peripheral Multiplexing register (PMUXn.PMUXE/O). The following code example shows how an I/O pin is multiplexed as LUT0 input 0 using ASF:

```
struct system_pinmux_config lut0_out_pin_conf;
system_pinmux_get_config_defaults(&lut0_out_pin_conf);
lut0_out_pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_OUTPUT;
lut0_out_pin_conf.mux_position = MUX_CCL_OUTPUT;
system_pinmux_pin_set_config(PIN_CCL_OUTPUT, &lut0_out_pin_conf);
```

Note: Not all LUTs can be connected to pins on the lower pin count devices.

1.1.3. Linked LUT Input (LINK)

The LUTs can be linked by applying a LUT output as input to a different LUT. Linking LUTs can for instance be used to create more complex waveforms, or simply to increase the number of available inputs. When LUTs are connected in cascade the number of inputs is 2N+1 where N is the number of connected LUTs. Linked LUT input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x2. The following table shows the corresponding inputs and outputs available for linking.

Table 1-1. LUT Linked Inputs

LUT	IN[x]	Output from LUT
LUT0	x = 0,1,2	LUT1
LUT1	x = 0,1,2	LUT2
LUT2	x = 0,1,2	LUT3
LUT3	x = 0,1,2	LUT0

1.1.4. Internal Feedback Input (FEEDBACK)

The output from the sequential logic block can be selected as input to the LUTs in the corresponding unit. Internal feedback input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x1.

1.1.5. Internal Events Input (EVENT)

Asynchronous events from Event System can be used as input selection. One event input line is available for each LUT and can be selected on each LUT input. Internal events input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x3. Before enabling the event selection, the Event System must be configured.

The CCL includes an edge detector which generates an internal strobe when the event rising edge is detected. The strobe duration is one GCLK_CCL clock cycle. To ensure the proper operation, the following steps must be followed:

- Enable the GCLK_CCL clock
- Configure Event System to route the desired event asynchronously
- Set the Inverted Event Input Enable bit in LUT Control register (LUTCTRLx.INVEI = 1) if a strobe must be generated on the event input falling edge
- Set the Event Input Enable bit in LUT Control register (LUTCTRLx.LUTEI = 1)

1.1.6. Analog Comparator Input (AC)

The AC outputs can be used as input source for the LUT. The AC compares voltage levels for two inputs, and provides a digital output based on this comparison. Each AC instance contains two comparators and each LUT can be connected to one designated comparator, shown in the following table. Analog comparator input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x5. Before selecting the comparator output, the AC must be configured.

Table 1-2. LUT AC Inputs

LUT	IN[x]	AC	CMP
LUT0	x = 0,1,2	AC0	CMP0
LUT1	x = 0,1,2		CMP1
LUT2	x = 0,1,2	AC1	CMP0
LUT3	x = 0,1,2		CMP1

1.1.7. Timer/Counter Inputs (TC and ALTTC)

The TC waveform output WO[0] can be used as input source. A total of two TCs can be applied as inputs to one LUT, default and alternative TC. TC waveform input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x6 or 0x7. For LUTx, TCx and the subsequent TC(x+1) instances are available as default and alternative TC selections. The corresponding LUTs and input TCs are indicated in the following table. Before selecting the TC waveform output, the TC must be configured.

Table 1-3. LUT TC Inputs

LUT	IN[x]	TC	ALTTC
LUT0	x = 0,1,2	TC0	TC1
LUT1	x = 0,1,2	TC1	TC2
LUT2	x = 0,1,2	TC2	TC3
LUT3	x = 0,1,2	TC3	TC4

Note: The following errata applies to Rev. A devices of SAM L21. The default TC selection is TC4/TC0/TC1/TC2 instead of TC0/TC1/TC2/TC3, and the alternate TC selection is TC0/TC1/TC2/TC3 instead of TC1/TC2/ TC3/ TC4. **Errata reference: 13406**.

1.1.8. Timer/Counter for Control Application Inputs (TCC)

The TCC waveform outputs WO[2:0] can be used as input source. TCC waveform input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x8. Each LUT input IN[x] corresponds to a specific TCC waveform output WO[x], implying that a total of three

TCC waveforms can be applied to each LUT. The corresponding LUTs and TCC waveform outputs are indicated in the following table. Before selecting the TCC waveform output, the TCC must be configured.

Table 1-4. LUT TCC Inputs

LUT	IN[x]	TCC	WO
LUT0	x = 0	TCC0	WO[0]
	x = 1		WO[1]
	x = 2		WO[2]
LUT1	x = 0	TCC1	WO[0]
	x = 1		WO[1]
	x = 2		WO[2]
LUT2	x = 0	TCC2	WO[0]
	x = 1		WO[1]
	x = 2		WO[2]
LUT3	x = 0	TCC3	WO[0]
	x = 1		WO[1]
	x = 2		WO[2]

1.1.9. Serial Communication Output Transmit Inputs (SERCOM)

The SERCOM transmitter output can be used as input source. The transmitter output is TxD for USART, MOSI for SPI, and SDA for I2C. SERCOM input is selected by writing the Input y Source Selection bit group in the LUT Control x register (LUTCTRLx.INSELy) to 0x9. LUTx can be connected to SERCOMx, as indicated in the following table. Before selecting the SERCOM as input source, the SERCOM must be configured.

Table 1-5. LUT SERCOM Inputs

LUT	IN[x]	SERCOM
LUT0	x = 0,1,2	SERCOM0
LUT1	x = 0,1,2	SERCOM1
LUT2	x = 0,1,2	SERCOM2
LUT3	x = 0,1,2	SERCOM3

1.2. Programmable Lookup Table

Each LUT has a truth table determining its output value based on the input values. All combinations of the three inputs IN[0], IN[1] and IN[2] are indicated in the below table, where each combination corresponds to one of the bits in Truth Table bit group in the LUT Control x register (LUTCTRLx.TRUTH). TRUTH must therefore be set according to the desired output for each combination of inputs. For instance if it is desired to generate a high output only when all three inputs are low or all three inputs are high, TRUTH[0] and TRUTH[7] must be written to one (LUTCTRLx.TRUTH written to 0x81). If only the first two inputs are of interest, and IN[2] is masked, the IN[2] column and the TRUTH[4:7] rows can be neglected since IN[2] in this case always will be zero.

Table 1-6. Truth Table of LUT

IN[2]	IN[1]	IN[0]	OUT
0	0	0	TRUTH[0]
0	0	1	TRUTH[1]
0	1	0	TRUTH[2]
0	1	1	TRUTH[3]
1	0	0	TRUTH[4]
1	0	1	TRUTH[5]
1	1	0	TRUTH[6]
1	1	1	TRUTH[7]

Using the above table, one can select specific combinations of inputs, or create standard logic gates. Examples for creating the most common three input logic gates are shown in the following table.

Table 1-7. Implementation of Logic Gates with Three Inputs

	AND	NAND	OR	NOR	XOR	XNOR
TRUTH[7:0]	0x80	0x7F	0xFE	0x01	0x16	0xE9

1.3. Filter and Edge Detector

By default, the LUT output is a combinatorial function of the LUT inputs. This may cause some short glitches when the inputs change value. These glitches can be removed by clocking through filters, if demanded by application needs. The Filter Selection bits in LUT Control register (LUTCTRLx.FILTSEL) define the synchronizer or digital filter options. When a filter is enabled, the output will be delayed by three to four GCLK_CCL cycles. One APB clock cycle after a LUT is disabled, all corresponding internal filter logic is cleared.

The edge detector can be used to generate a pulse when detecting a rising edge on its input. To detect a falling edge, the truth table should be programmed to provide the opposite levels. The edge detector is enabled by writing a one to the Edge Selection bit in LUT Control register (LUTCTRLx.EDGESEL). In order to avoid unpredictable behavior, a valid filter option must be enabled as well. Refer to the Filter Selection bit group in the LUT Control x register (LUTCTRLx.FILTSEL) for valid filter selections.

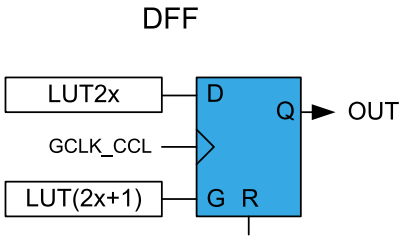
1.4. Sequential Logic

Each LUT pair can be connected to internal sequential logic: D flip flop, JK flip flop, gated D-latch or RS-latch can be selected by writing the corresponding Sequential Selection bits in Sequential Control x register (SEQCTRLx.SEQSEL). Before using sequential logic, the GCLK clock and optionally each LUT filter or edge detector, must be enabled.

Gated D Flip-Flop (DFF)

When the DFF is selected, the D-input is driven by the even LUT output (LUT2x), and the G-input is driven by the odd LUT output (LUT2x+1), as shown in Figure 1-2 D Flip Flop on page 8.

Figure 1-2. D Flip Flop



When the even LUT is disabled (LUTCTRL2x.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK_CCL, as shown in Table 1-8 DFF Characteristics on page 8.

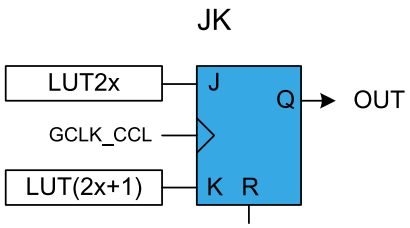
Table 1-8. DFF Characteristics

R	G	D	OUT
1	X	X	Clear
0	1	1	Set
		0	Clear
	0	X	Hold state (no change)

JK Flip-Flop (JK)

When this configuration is selected, the J-input is driven by the even LUT output (LUT2x), and the K-input is driven by the odd LUT output (LUT2x+1), as shown in Figure 1-3 JK Flip Flop on page 8.

Figure 1-3. JK Flip Flop



When the even LUT is disabled (LUTCTRL2x.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK_CCL, as shown in Table 1-9 JK Characteristics on page 8.

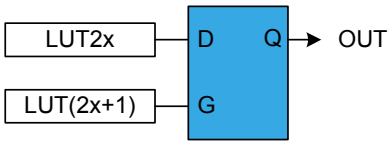
Table 1-9. JK Characteristics

R	J	K	OUT
1	X	X	Clear
0	0	0	Hold state (no change)
0	0	1	Clear
0	1	0	Set
0	1	1	Toggle

Gated D-Latch (DLATCH)

When the DLATCH is selected, the D-input is driven by the even LUT output (LUT2x), and the G-input is driven by the odd LUT output (LUT2x+1), as shown in Figure 1-2 D Flip Flop on page 8.

Figure 1-4. D-Latch



When the even LUT is disabled (LUTCTRL2x.ENABLE=0), the latch output will be cleared. The G-input is forced enabled for one more APB clock cycle, and the D-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in Table 1-10 D-Latch Characteristics on page 9.

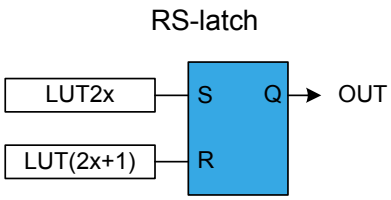
Table 1-10. D-Latch Characteristics

G	D	OUT
0	X	Hold state (no change)
1	0	Clear
1	1	Set

RS Latch (RS)

When this configuration is selected, the S-input is driven by the even LUT output (LUT2x), and the R-input is driven by the odd LUT output (LUT2x+1), as shown in Figure 1-5 RS-Latch on page 9.

Figure 1-5. RS-Latch



When the even LUT is disabled (LUTCTRL2x.ENABLE=0), the latch output will be cleared. The R-input is forced enabled for one more APB clock cycle and S-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in Table 1-11 RS-latch Characteristics on page 9.

Table 1-11. RS-latch Characteristics

S	R	OUT
0	0	Hold state (no change)
0	1	Clear
1	0	Set
1	1	Forbidden state

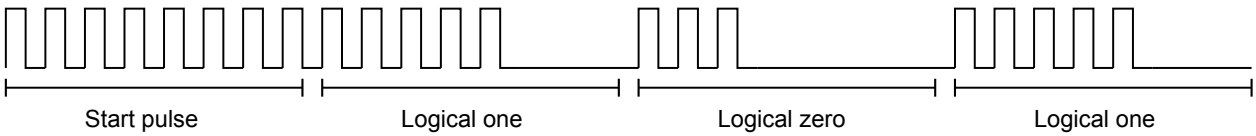
2. Example Overview

CCL can be used as a simple logic block for pin logic. A code example with three I/O pin inputs can be found in the Quick Start Guide for CCL in ASF. In the following application example a more advanced use case is presented. This example shows how CCL can be used with TCs, together with Event System and DMAC, to generate an IR encoded signal. The project developed using ASF is available for download as an attachment, along with this Application Note. Before the example is presented, it is necessary to have a brief introduction to IR encoding.

IR Encoding

There are many different standards for IR encoding. Most of them involve a fixed carrier frequency which will be on for a certain period and then off for a certain period. The enabling and disabling of the carrier frequency is determined by the dutycycle of a waveformed modulation signal. The dutycycle of the modulation signal will then determine if the transmitting value is a start pulse, a logical one or a logical zero. E.g. if the carrier frequency is signaled 3/4 of the period, this indicates a logical one, while 1/4 indicates a logical zero. This is illustrated in the following figure.

Figure 2-1. IR Encoded Signal



In the following example, no specific encoding will be applied but a more general approach where period and dutycycle can be adjusted is presented.

Implementation

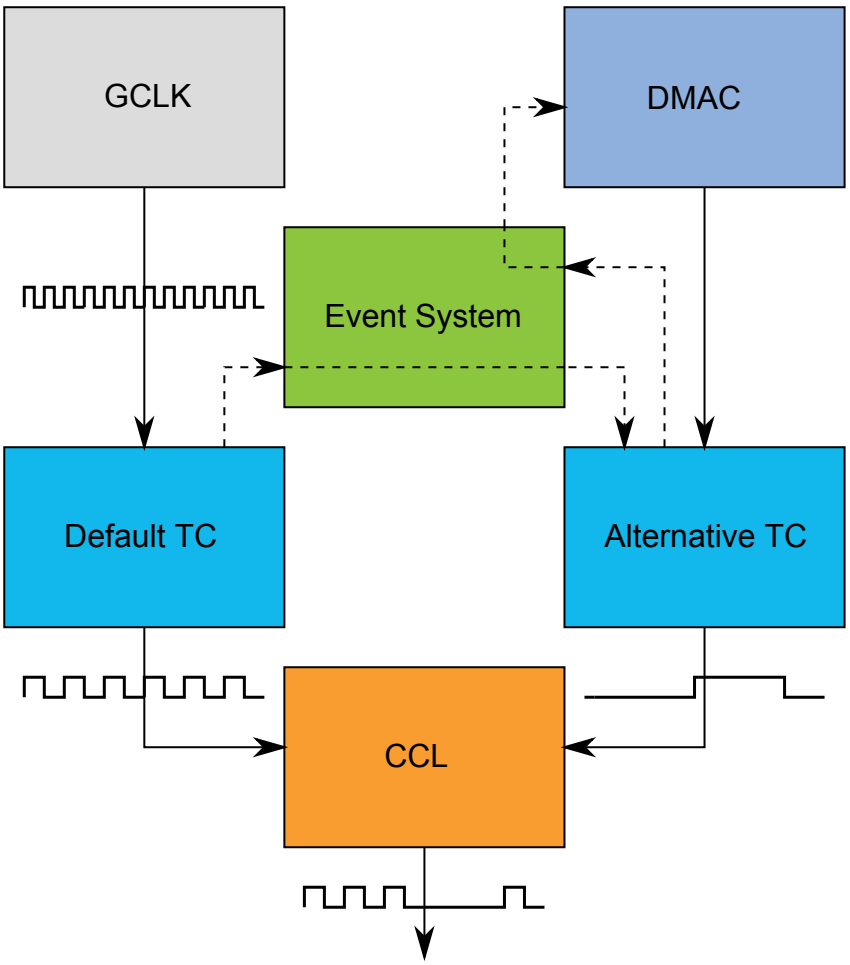
To generate a modulated IR signal, CCL is configured with two TC inputs while the last input is masked. The default TC is used to generate the carrier frequency of the IR signal while the alternative TC is applied as the modulation signal. To ensure the modulation signal is synchronized with the carrier frequency, the alternative TC increments on overflow events generated by the default TC. In the example, the truth table is written to 0x01. Referring to the Table 1-6 Truth Table of LUT on page 7, this corresponds to a high output when alternative TC waveform is low and default TC waveform is high. By adjusting the compare value for the alternative TC, the dutycycle will change, resulting in longer or shorter intervals of transmitting the carrier frequency.

To update compare value and period, the DMAC will be setup with two channels transferring data from two arrays in memory to the dedicated TC registers. Using DMAC to update the compare value register and the peroid register allows the device to operate from Standby sleep mode.

An illustration of the application flow is shown in Figure 2-2 Schematic of Implementation on page 11.

In the following sections, only code for CCL configuration is attached. The configuration of TC, DMAC and EVSYS will be described but the implementation is only available in the example project enclosed to this application note.

Figure 2-2. Schematic of Implementation



2.1. Prerequisites

Executing the example described in this application note requires:

- Atmel Studio 7 or later
- Atmel Software Framework 3.27 or later
- SAM L21/L22 Xplained Pro Evaluation Kit with USB cable

2.2. Peripherals

The following peripherals are used in this example:

- Configurable Custom Logic (CCL)
- Direct Memory Access Controller (DMAC)
- Event System (EVSYS)
- Timer/Counter (TC)

2.3. Configurations

2.3.1. Configure CCL

The CCL must be configured to the desired logical functionality. The `configure_ccl_lut0()` function initializes the CCL module with the desired clock generator and configures LUT0 according to the following settings:

- Input 0 is default TC
- Input 1 is alternative TC
- Input 2 is masked
- Truth table value set to 0x01
- Run in Standby enabled

In addition, the encoded signal is multiplexed to the I/O pin corresponding to CCL0 output. The configuration function follows the ASF standard by creating a configuration struct, loading default settings, applying necessary changes, and enabling the module.

```
void configure_ccl_lut0(void)
{
    /** Creates a new configuration structure for CCL. */
    struct ccl_config conf;

    /** Apply the default settings. */
    ccl_get_config_defaults(&conf);
    conf.clock_source   = GCLK_GENERATOR_1;
    conf.run_in_standby = true;

    /** Initialize CCL with the user settings. */
    ccl_init(&conf);

    /** Creates a new configuration structure for LUT0. */
    struct ccl_lut_config conf_ccl_lut0;

    /** Apply the default settings. */
    ccl_lut_get_config_defaults(&conf_ccl_lut0);

    /** Configure LUT0. */
    conf_ccl_lut0.truth_table_value = 0x01;
    conf_ccl_lut0.input0_src_sel    = CCL_LUT_INPUT_SRC_TC;
    conf_ccl_lut0.input1_src_sel    = CCL_LUT_INPUT_SRC_ALTTC;
    conf_ccl_lut0.input2_src_sel    = CCL_LUT_INPUT_SRC_MASK;

    /** Set up LUT0 output pin. */
    struct system_pinmux_config lut0_out_pin_conf;
    system_pinmux_get_config_defaults(&lut0_out_pin_conf);
    lut0_out_pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_OUTPUT;
    lut0_out_pin_conf.mux_position = CCL_OUTPUT_MUX;
    system_pinmux_pin_set_config(CCL_OUTPUT_PIN, &lut0_out_pin_conf);

    /** Initialize and enable LUT0 with the user settings. */
    ccl_lut_set_config(CCL_LUT_0, &conf_ccl_lut0);

    /** Enable CCL module. */
    ccl_lut_enable(CCL_LUT_0);
    ccl_module_enable();
}
```

2.3.2. Configure TC

The two TCs used as inputs for LUT0 are configured in `configure_tc_def()` and `configure_tc_alt()`. The default TC is configured to generate the carrier frequency while the

alternative TC is used as the modular signal. Both TCs are configured as 8 bit counters with PWM generation and events enabled. The default TC is configured to generate event on overflow while the alternative TC is configured to generate event on overflow and to increment counter on incoming event. For comparison of CCL output signal and the TC waveform outputs, both TCs are multiplexed to I/O pins.

2.3.3. Configure DMAC

To use DMAC for transferring data it is necessary to configure a DMA resource and transfer descriptor. The resource contains the peripheral trigger, the trigger action and more channel configurations. The descriptor contains information regarding the specific data transfer, such as source and destination address, data size, next descriptor address, transfer counter and more. In the example, two different channels are used to write to the TC registers. For this purpose, two resources and two descriptors are needed. The resources are configured with the `configure_dma_resource()` function while the descriptors are configured using the `configure_dma_channel_x()` functions. Both DMA channels are configured to run in standby with alternative TC overflow as trigger. The next descriptor address field is used to loop the descriptors back to start, to enable the message to be sent repeatedly. The first channel is used for updating the alternative TC period. The source address is set to the array containing the different periods and the destination is the period register. The second channel is used for updating the alternative TC compare value register. The source address is set to the array containing the different compare values and the destination is the compare value register.

2.3.4. Configure EVSYS

Events must be enabled as generators and/or triggers when configuring peripherals. In this example, event generation on overflow is enabled for both default and alternative TC. The default TC is also set to count on incomming events and DMAC is set to trigger a transfer on incoming events. In addition, the event system must be configured to link event generators with event users. In `configure_tc_evsys()` the alternative TC is set as event user since this counter should increment incoming events. The event generator is overflow on the default TC. In `configure_dma_evsys()` the DMAC is configured to start a transfer on alternative TC overflow event. DMAC channel 0 and 1 are therefore set as event users while the event generator is alternative TC overflow. With this setup, the alternative TC period and compare value registers will be updated after every periodic overflow.

2.3.5. Edit Configuration Files

The clock system is configured in the included `conf_clocks.h` header file. To enable Standby sleep mode operation, the selected clock source is set to run in standby. The peripheral clock source, generic clock generator 1, is enabled. This clock generator offers a wide range of prescale values. Example specific defines are placed in the included header file `conf_ccl_example.h`. The defines enables the user to switch default and alternative TCs, depending on device revision. This configuration file also controls the I/O pins and events used in the application.

2.4. Main Routine

In the `main()` function, all the previous configuration functions are executed before the TCs are enabled. To lower the power consumption, the device is set to enter Standby sleep mode. Since peripherals and clock sources are set to run in standby, the device will continue to generate the encoded signal while sleeping.

3. Software License

4. Revision History

Doc Rev.	Date	Comments
42419B	01/2016	Updated for SAM L22 device
42419A	03/2015	Initial document release

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**