TB3218

Getting Started with Configurable Custom Logic (CCL)

Introduction

Author: Cristian Pop, Microchip Technology Inc.

The Configurable Custom Logic (CCL) is a programmable logic peripheral that allows the on-chip creation of the logic functions for Microchip tinyAVR® 0- and 1-series, megaAVR® 0-series, and AVR® DA devices. The CCL provides programmable, combinational and sequential logic that operates independently of the CPU execution. It can be connected to a wide range of internal and external inputs such as device pins, events, or other internal peripherals and can serve as a “glue logic” between the device peripherals and external devices.

This technical brief explains how to use the CCL to implement the following functions:

- **Logic AND Gate:**
  Uses CCL to implement a simple logic AND gate.
- **State Decoder:**
  Shows how to use CCL combinational logic to detect a specific state of the external signals.
- **SR Latch:**
  Uses internal CCL sequential logic to create an SR latch.

Note: For each of the use cases described in this document, there are two code examples: One bare metal developed on ATmega4809, and one generated with MPLAB® Code Configurator (MCC) developed on AVR128DA48.

View the ATmega4809 Code Example on GitHub
Click to browse repository

View the AVR128DA48 Code Example on GitHub
Click to browse repository

Note: In addition to the use cases mentioned above, this document provides advanced examples for the CCL peripheral, as described in 6. Advanced Examples. These are generated with MCC and developed on AVR128DA48.

Table of Contents

1. Relevant Devices

This section lists the relevant devices for this document. The following figures show the different family devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features. Downward migration on tinyAVR® 1-series devices may require code modification due to fewer available instances of some peripherals.
- Horizontal migration to the left reduces the pin count and, therefore, the available features
- Devices with different Flash memory sizes typically also have different SRAM and EEPROM

Figure 1-1. tinyAVR® 0-series Overview

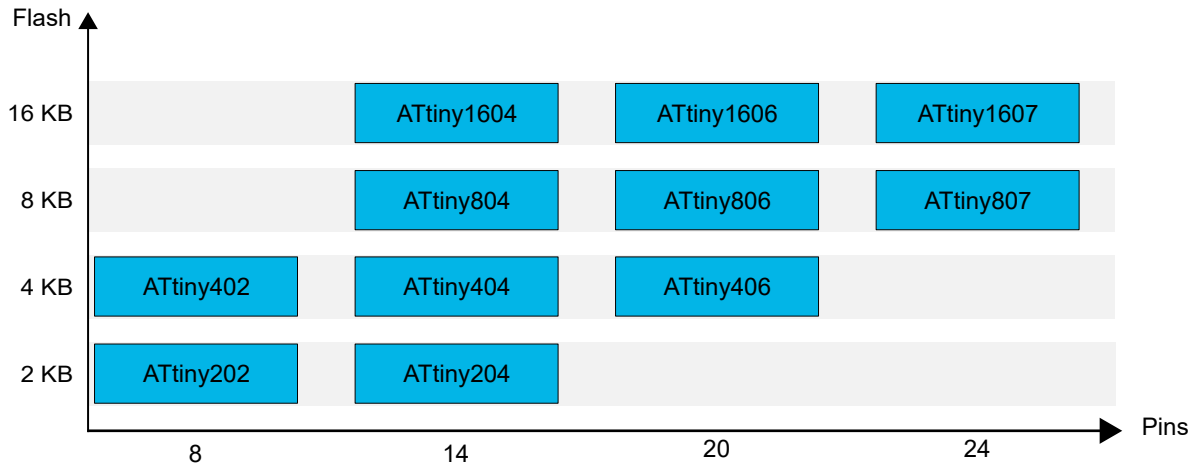


Figure 1-2. tinyAVR® 1-series Overview

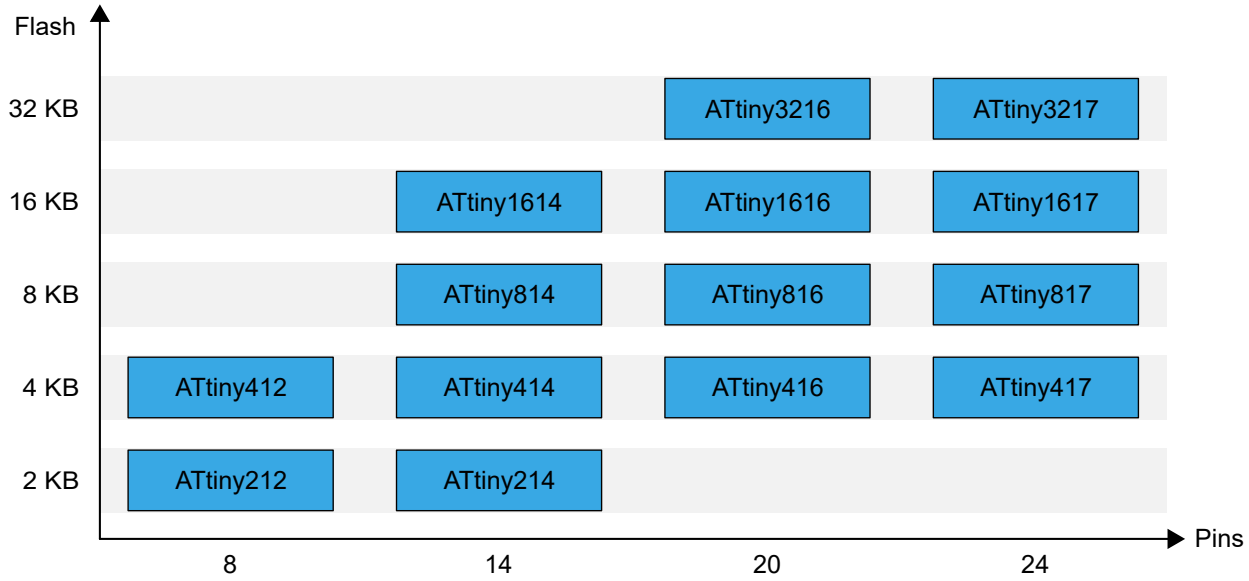


Figure 1-3. tinyAVR® 2 Family Overview

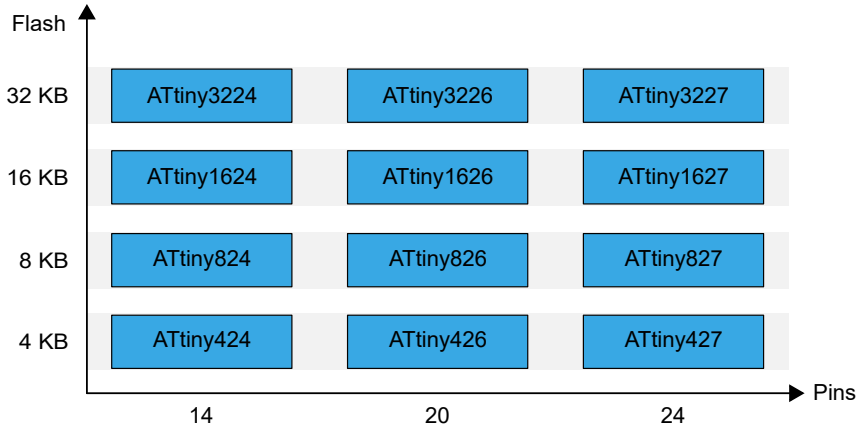


Figure 1-4. megaAVR® 0-series Overview

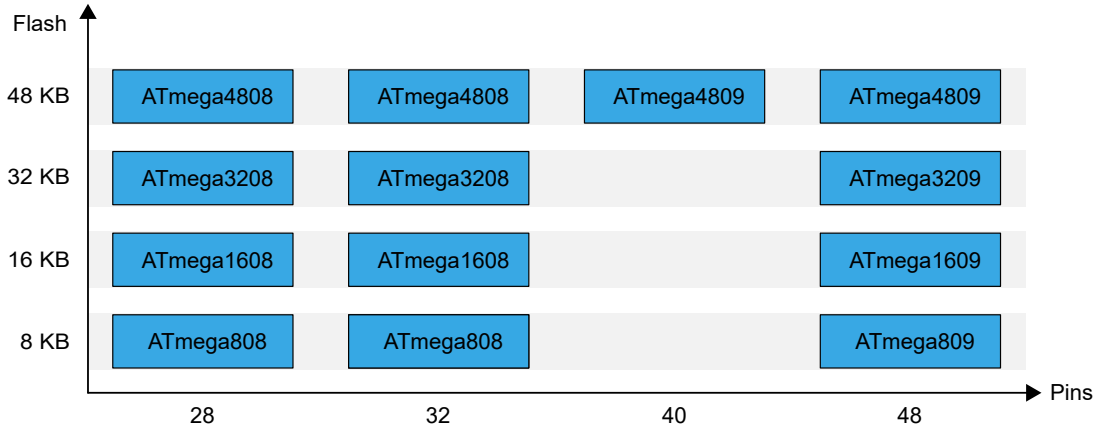


Figure 1-5. AVR® DA Family Overview

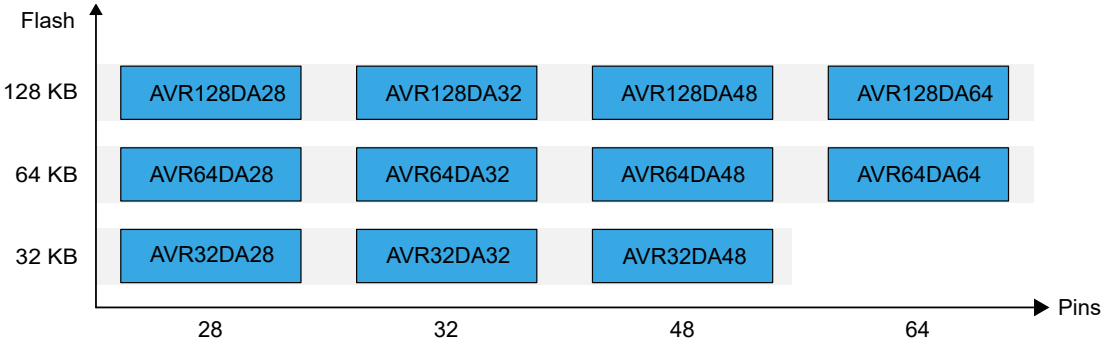
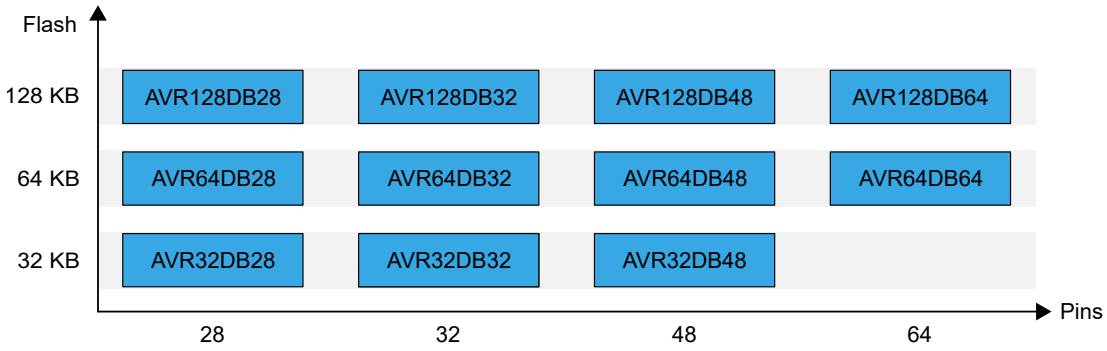


Figure 1-6. AVR® DB Family Overview



2. Overview

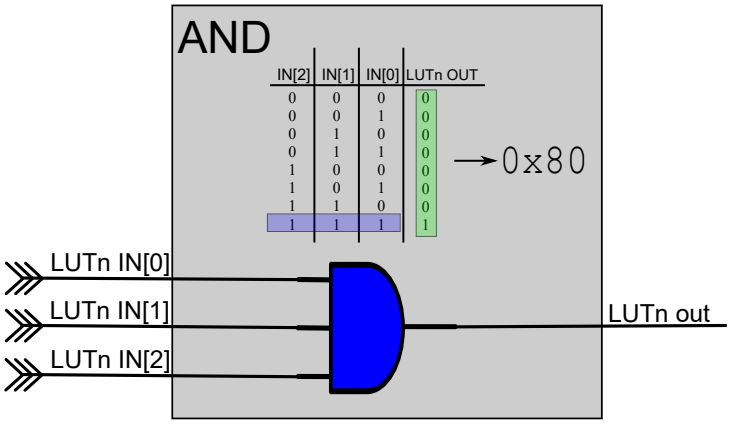
The CCL peripheral has one pair of Look-up Tables (LUTs) on tinyAVR 0- and 1-series, two pairs of LUTs on megaAVR 0-series, and three pairs of LUTs on AVR DA devices. Each LUT consists of three inputs with a truth table, a synchronizer, a filter, and an edge detector. Each LUT can generate an output as a user-programmable logic expression with three inputs; any device with CCL will have a minimum of two LUTs available. These inputs can be individually masked. The output can be generated from the combinatorial inputs and be filtered to remove spikes. An optional sequential logic module can be enabled. The inputs to the sequential module are individually controlled by two independent, adjacent LUT outputs (LUT0/LUT1), enabling complex waveform generation.

Truth Table

It is possible to create simple logic blocks (AND, OR, NAND, NOR, XOR) or custom ones using the truth table up to three inputs on each of the LUTs. When more than three inputs are required, multiple connected LUTs are used to create logic gates.

To define a combinational specific logic function, the CCL module uses truth tables. A truth table shows how the logic circuit responds to various combinations of three inputs. Each combination of the Input (IN[2:0]) bits corresponds to one bit in the respective TRUTHn register. Below are some examples of how to create some common logic gates using three inputs.

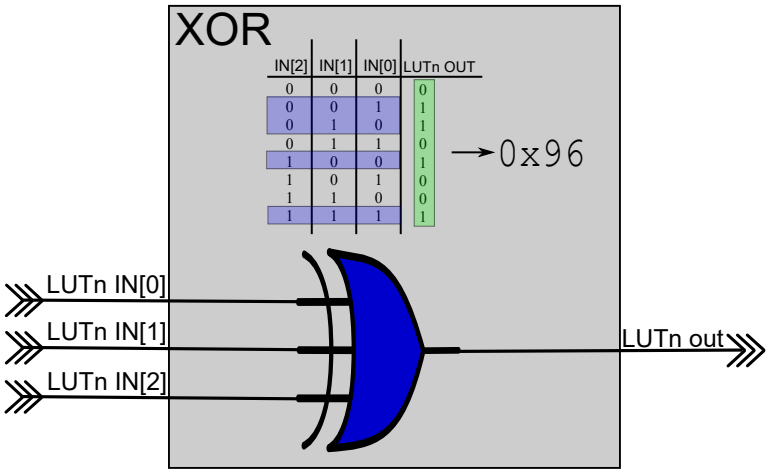
Figure 2-1. AND Logic



To get a HIGH(1) output from an AND gate, all inputs must be HIGH(1). Looking at the truth table above, only TRUTH[7] fulfills this requirement if all three inputs are used. This means that TRUTH[7] must be HIGH(1) and the rest must be LOW(0), resulting in the hex value of `0x80` to be used in the TRUTHn register.

```
CCL.TRUTHn = 0x80;
```

Figure 2-2. XOR Gate



To get a HIGH(1) output from an XOR gate, the number of HIGH(1) inputs must be odd. Looking at the truth table above, TRUTH[1], TRUTH[2], TRUTH[4], and TRUTH[7] fulfill this requirement. This means that these bits must be HIGH(1) and the rest must be LOW(0), resulting in the hex value of `0x96` to be used in the TRUTHn register.

```
CCL.TRUTHn = 0x96;
```

When any of the three inputs are not needed, the unused input will be masked (tied low). Only the TRUTH bits where the masked input is '0' can be used when looking at the truth table to determine how the bits need to be set to get the wanted logic. Below are some examples of where various inputs are masked.

Figure 2-3. Two-Input AND Gate, IN[0] Masked

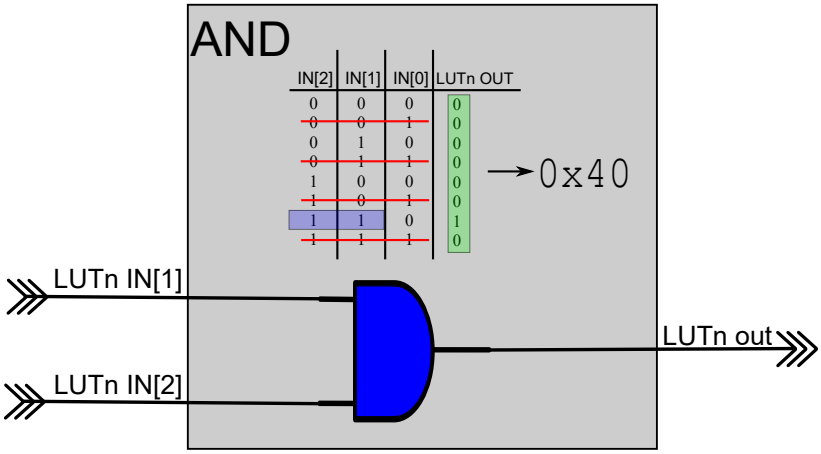
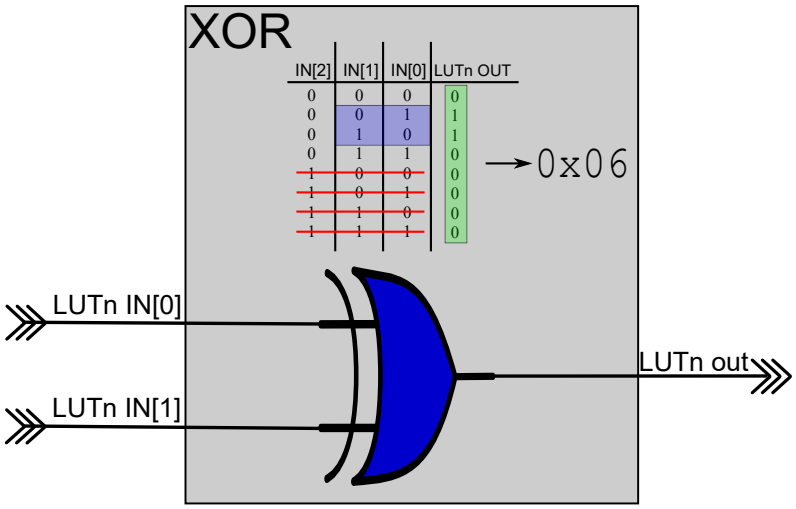


Figure 2-4. Two-Input XOR Gate, IN[2] Masked



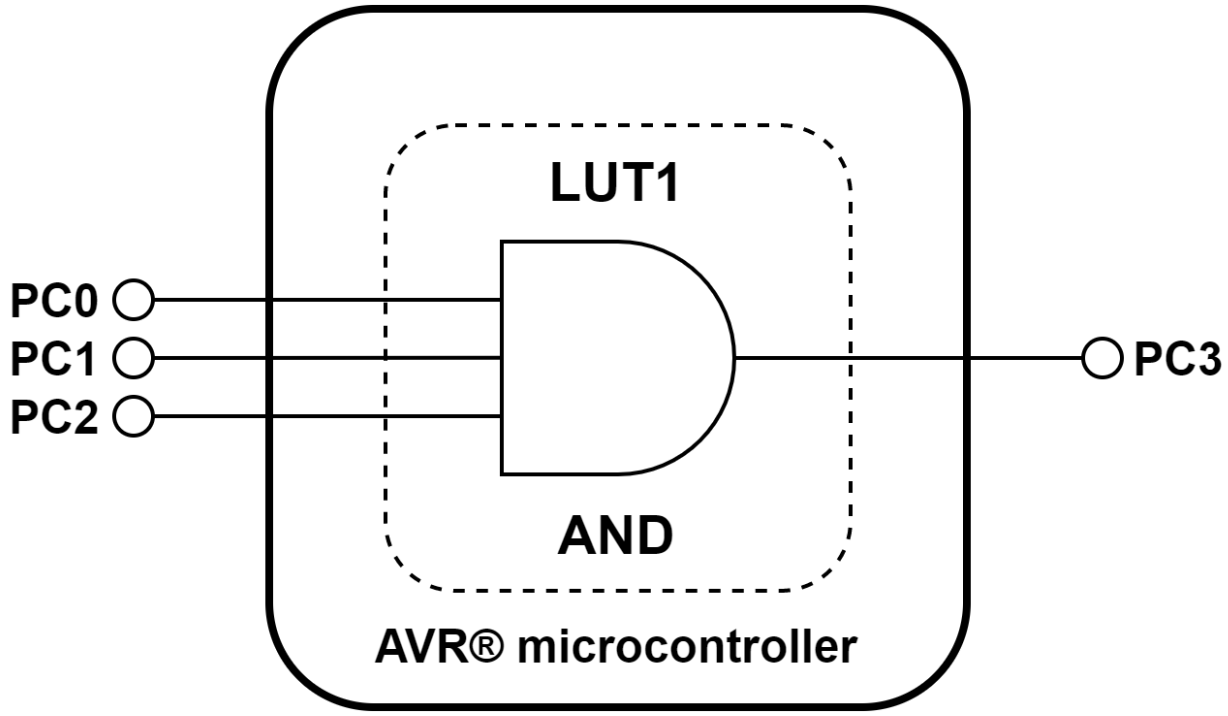
Some applications require more than three logic inputs. The CCL module provides the option to link internally the next LUTs direct output to a LUT input. For example, if LUT0 and LUT1 are used to create a logic function, the LUT1 output can be connected to the LUT0 input internally.

Using the CCL eliminates the need for external logic, reduces Bill of Materials (BOM) cost, and enables the CPU to handle time-critical parts of the application more efficiently.

3. Logic AND Gate

The following example shows how to configure and use CCL LUT1 to implement an AND gate with three inputs.

Figure 3-1. Using CCL as Logic AND Gate



The first step is to select the I/O pins as inputs using the INSELx[3:0] bits from the LUT Control (LUTnCTRLB and LUTnCTRLC) registers.

Figure 3-2. LUTn Control B Register

Bit	7	6	5	4	3	2	1	0
	INSEL1[3:0]				INSEL0[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 3-3. LUTn Control C Register

Bit	7	6	5	4	3	2	1	0
					INSEL2[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

The table below summarizes the INSEL[3:0] options for all inputs.

Figure 3-4. CCL Input Selection Options

Value	Input Source	INSEL0	INSEL1	INSEL2
0x00	MASK	None		
0x01	FEEDBACK	LUTn		
0x02	LINK	LUT(n+1)		
0x03	EVENT0	Event input source 0		
0x04	EVENT1	Event input source 0		
0x05	IO	IN0	IN1	IN2
0x06	AC	AC0 OUT		
0x07	-			
0x08	USART	USART0 TXD	USART1 TXD	USART2 TXD
0x09	SPI	SPI0 MOSI	SPI0 MOSI	SPI0 SCK
0x0A	TCA0	WO0	WO1	WO2
0x0B	-			
0x0C	TCB	TCB0 WO	TCB1 WO	TCB2 WO
Other	-			

This translates to the following code.

```
CCL.LUT1CTRLB = CCL_INSEL0_IO_gc | CCL_INSEL1_IO_gc;
CCL.LUT1CTRLC = CCL_INSEL2_IO_gc;
```

The next step is to configure the truth tables for LUT1 to generate the right combinational logic to implement an AND gate on the selected pins. Thus, the truth table will have a value of `0x80`.

```
CCL.TRUTH1 = 0x80;
```

The next step is to configure the output of the decoder, specifically, the I/O Port pin (PC3) in this example. This is done by setting the OUTEN bit on the LUT0CTRLA register.

Figure 3-5. LUTn Control A Register

Bit	7	6	5	4	3	2	1	0
	EDGEDET	OUTEN	FILTSEL[1:0]		CLKSRC[2:0]			ENABLE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This translates to the following code:

```
CCL.LUT1CTRLA = CCL_OUTEN_bm;
```

By enabling the LUTn output on the I/O pin, the settings for the corresponding pin are overwritten. To enable the decoding of the input sequence, the CCL and the used LUTs need to be enabled. That is done using the ENABLE bit from the LUTnCTRLA register.

```
CCL.LUT1CTRLA |= CCL_ENABLE_bm;
```

To complete the setup, the CCL module needs to be enabled using a CCL Global Enable bit from the CTRLA register.

Figure 3-6. CCL Control A Register

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY						ENABLE
Access		R/W						R/W
Reset		0						0

```
CCL.CTRLA = CCL_ENABLE_bm;
```

Tip: The full code example is also available in the Appendix section.

View the ATmega4809 Code Example on GitHub
Click to browse repository

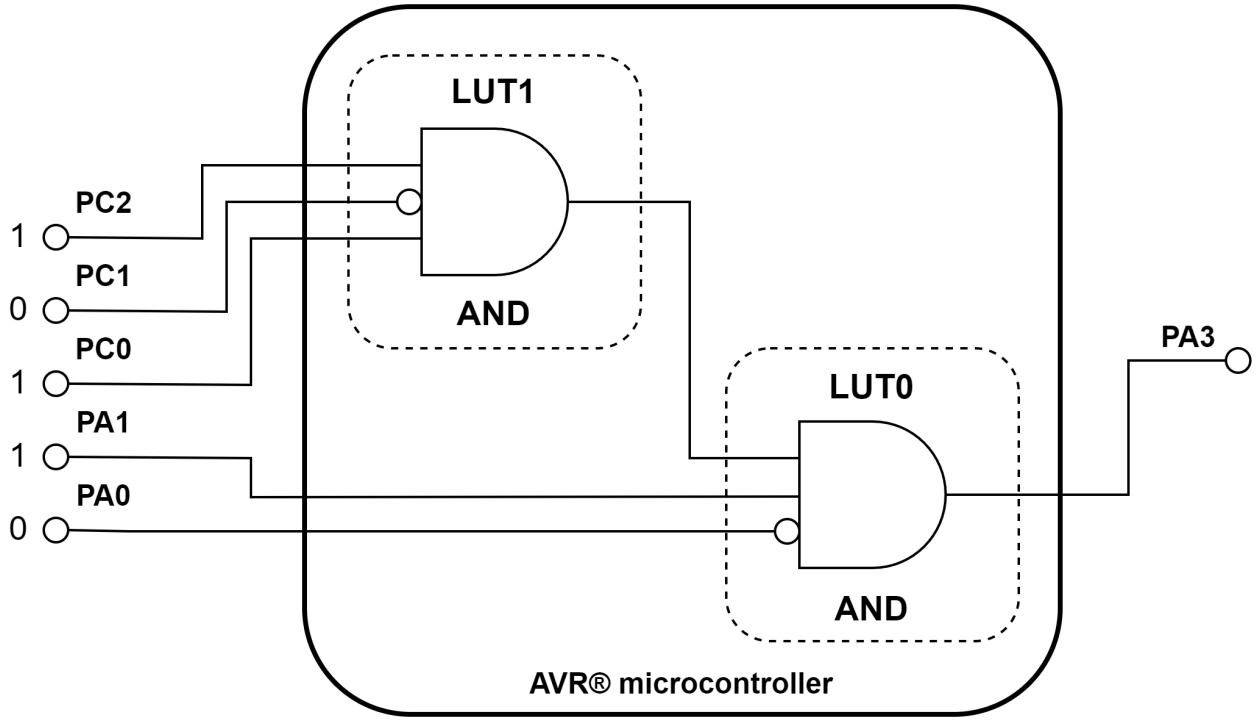
An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:

View the AVR128DA48 Code Example on GitHub
Click to browse repository

4. State Decoder

The application may need to detect when a specific combination of signals (pattern) appears on the pins. By combining logic gates, a simple state decoder for external signals can be implemented without involving the CPU.

Figure 4-1. Using the AVR® Microcontroller as a State Decoder



In this example, the CCL module will be used to decode the presence of the `b'10110` pattern on the input pins. LUT0 and LUT1, connected to the corresponding input pins, will be used.

The input selection from different input options is done using the INSELx[3:0] bits from the LUT Control (LUTnCTRLB and LUTnCTRLC) registers, as shown in the following figures.

Figure 4-2. LUTn Control B Register

Bit	7	6	5	4	3	2	1	0
	INSEL1[3:0]				INSEL0[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 4-3. LUTn Control C Register

Bit	7	6	5	4	3	2	1	0
					INSEL2[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

The table below summarizes the INSEL[3:0] options for all inputs.

Figure 4-4. CCL Input Selection Options

Value	Name	Description
0x0	MASK	None (masked)
0x1	FEEDBACK	Feedback input
0x2	LINK	Output from LUTn+1
0x3	EVENT0	Event input source 0
0x4	EVENT1	Event input source 1
0x5	IO	I/O-pin LUTn-IN0
0x6	AC0	AC0 out
0x7	-	Reserved
0x8	USART2	USART2 TXD
0x9	SPI0	SPI0 SCK
0xA	TCA0	TCA0 WO1
0xB	-	Reserved
0xC	TCB2	TCB2 WO
Other	-	Reserved

For this example, two adjacent LUTs (LUT0 and LUT1) will be used, with the output of LUT1 connected to the LUT0 input (linked).

```
CCL.LUT0CTRLC = CCL_INSEL2_LINK_gc;
```

The other two inputs of LUT0 and all three inputs of LUT1 are connected to the I/O pins.

```
CCL.LUT0CTRLB = CCL_INSEL0_IO_gc | CCL_INSEL1_IO_gc;
CCL.LUT1CTRLB = CCL_INSEL0_IO_gc | CCL_INSEL1_IO_gc;

CCL.LUT1CTRLC = CCL_INSEL2_IO_gc;
```

The following step is to configure the truth tables for LUT0 and LUT1 to generate the right combinational logic to detect `b'10110` on the selected pins. The TRUTH1 table is used to decode the pattern for the Most Significant three bits (`b'10110`).

```
CCL.TRUTH1 = 0x20;
```

LUT0 has as inputs two Least Significant bits from the input pattern (`b'10110`) and the decoded output of LUT1 on the third input, resulting in binary sequence `b'110` to be decoded. The value of the truth table, in this case, will be `0x40`.

```
CCL.TRUTH0 = 0x40;
```

The next step is to configure the output of the decoder, specifically, the I/O PORT pin PA3 in this example. This is done by setting the OUTEN bit on the LUT0CTRLA register.

Figure 4-5. LUTn Control A Register

Bit	7	6	5	4	3	2	1	0
	EDGEDET	OUTEN	FILTSEL[1:0]		CLKSRC[2:0]			ENABLE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

This translates to the following code.

```
CCL.LUT0CTRLA = CCL_OUTEN_bm;
```

By enabling the LUTn output on the I/O pin, the settings for the corresponding pin are overwritten. To complete the setup and the start decoding of the input sequence, the CCL and used LUTs need to be enabled. That is done using the ENABLE bit from the LUTnCTRLA register.

```
CCL.LUT1CTRLA = CCL_ENABLE_bm;
CCL.LUT0CTRLA |= CCL_ENABLE_bm;
```

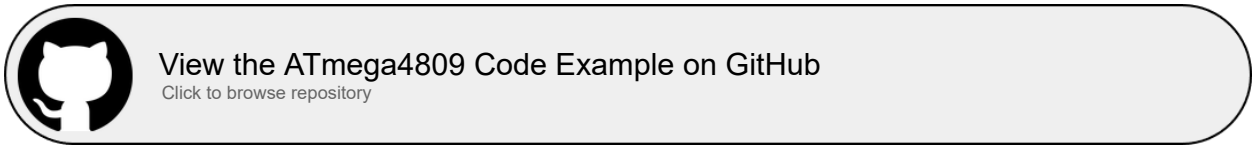
To complete the setup, the CCL module needs to be enabled using a CCL Global Enable bit from the CTRLA register.

Figure 4-6. CCL Control A Register

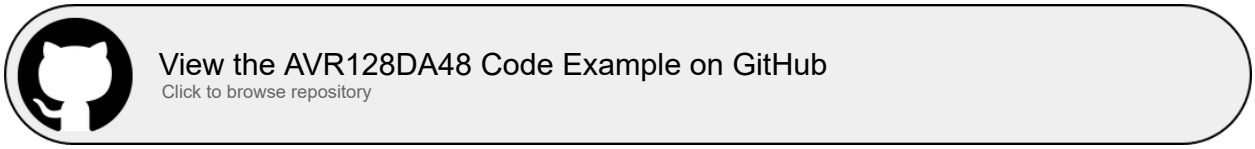
Bit	7	6	5	4	3	2	1	0
		RUNSTDBY						ENABLE
Access		R/W						R/W
Reset		0						0

```
CCL.CTRLA = CCL_ENABLE_bm;
```

Tip: The full code example is also available in the Appendix section.



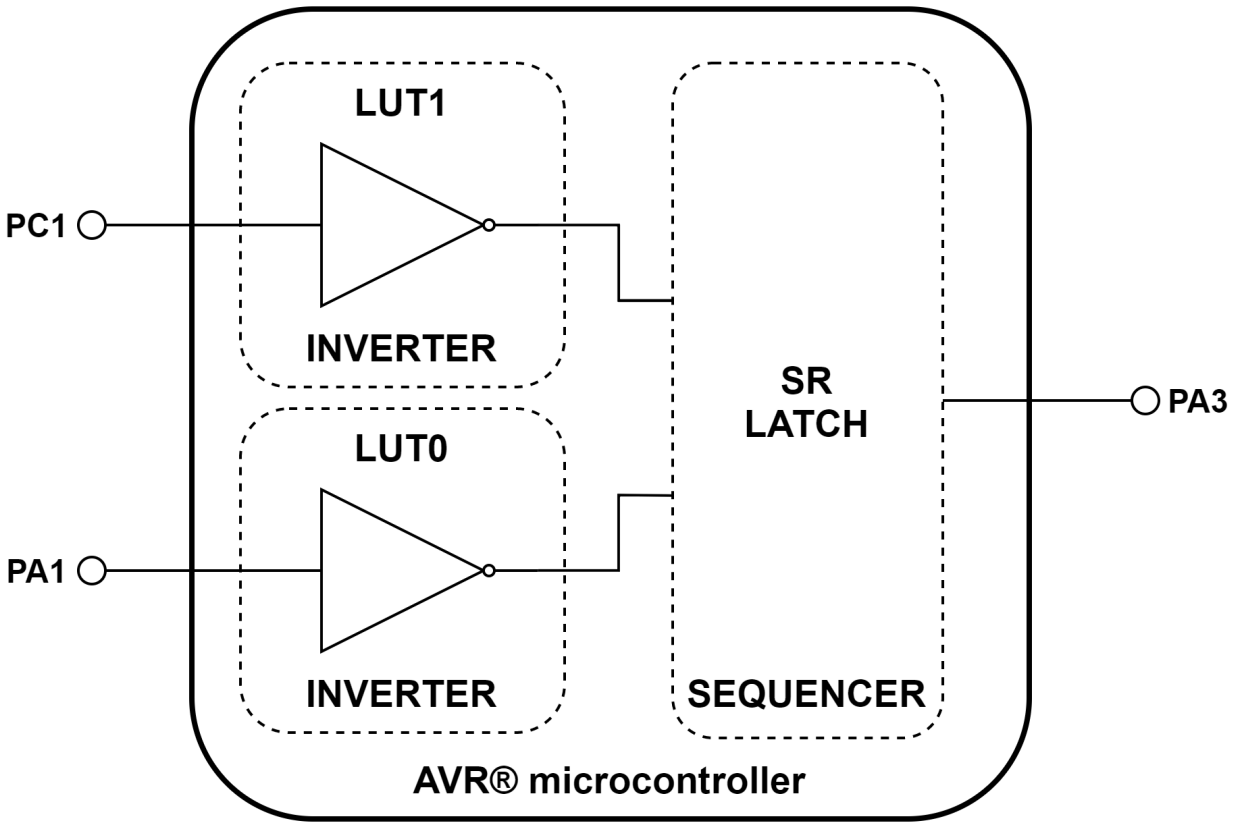
An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:



5. SR Latch

This section describes an application example that uses CCL combinational and sequential logic to implement an SR latch. This functionality can be created using two adjacent LUTs (LUT0 and LUT1) connected through a sequential logic block.

Figure 5-1. Using CCL to Implement an SR Latch



For Set and Reset signals, two pins are used as inputs for LUTs (I/O PORT pin PA1 and I/O PORT pin PC1). That translates to the following code.

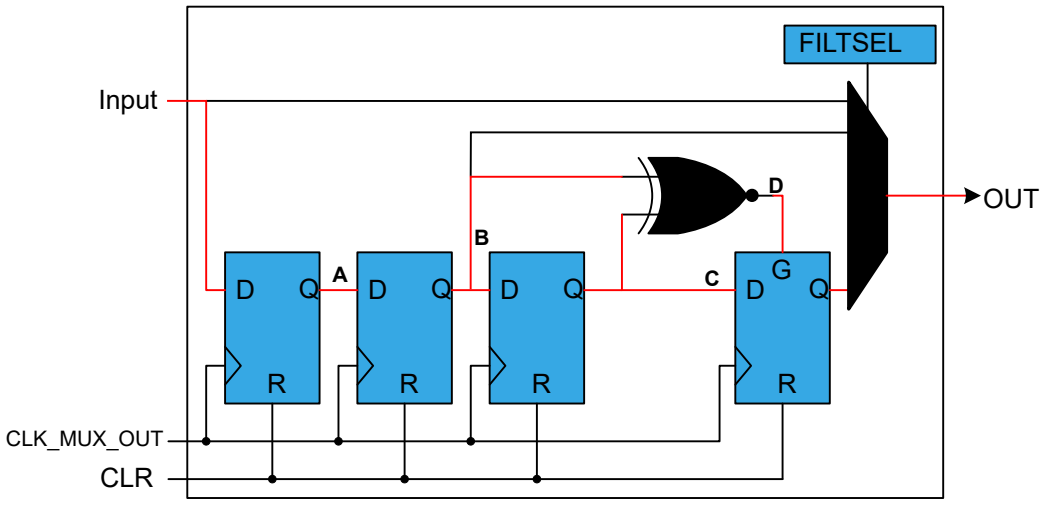
```
CCL.LUT0CTRLB = CCL_INSEL0_MASK_gc | CCL_INSEL1_IO_gc;
CCL.LUT0CTRLC = CCL_INSEL2_MASK_gc;
CCL.LUT1CTRLB = CCL_INSEL0_MASK_gc | CCL_INSEL1_IO_gc;
CCL.LUT0CTRLC = CCL_INSEL2_MASK_gc;
```

In this case, only the input selected for the Input signal needs to be considered when configuring the Truth register for each LUT. For instance, if the signal is active-high and available on LUTn_IN[1], the Truth register will be set to `0x02`. If the input signal is active-low, which is the case for many evaluation kits, the Truth register will be set to `0x01`. For the selected example, the input signals are active-low, so the Truth register will be set to `0x01` for both LUTs.

```
CCL.TRUTH0 = 0x01;
CCL.TRUTH1 = 0x01;
```

The truth table output is a combinatorial function of the inputs. This may cause some short glitches when the inputs change value. These glitches may not cause any problems, but if the LUT output is set to trigger an event, used as input on a timer or similar, an unwanted glitch may trigger unwanted events and peripheral action. In removing these glitches by clocking through the filters, the user will only get the intended output. Each Look-up Table (LUT) in the CCL includes a filter that can be used to synchronize or filter the LUT output.

Figure 5-2. CCL Filter



The selection of the filter option is done by using the FILTSEL[1:0] bits from the LUTnCTRLA register.

Figure 5-3. CCL Filter Options

Value	Name	Description
0x0	DISABLE	Filter disabled
0x1	SYNCH	Synchronizer enabled
0x2	FILTER	Filter enabled
0x3	-	Reserved

```
CCL.LUT0CTRLA = CCL_FILTSEL_FILTER_gc;
CCL.LUT1CTRLA = CCL_FILTSEL_FILTER_gc;
```

The next step is to connect the LUTs through a sequential logic to create SR latch functionality. The bits in SEQSEL0[3:0] from the Sequential Control (SEQCTRL0) register select the sequential configuration for LUT0 and LUT1.

Figure 5-4. Sequential 1 Control 0 Register

Bit	7	6	5	4	3	2	1	0
					SEQSELn[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – SEQSELn[3:0]: Sequential Selection bits
The bits in SEQSELn select the sequential configuration for LUT[2n] and LUT[2n+1].

Value	Name	Description
0x0	DISABLE	Sequential logic is disabled
0x1	DFF	D flip flop
0x2	JK	JK flip flop
0x3	LATCH	D latch
0x4	RS	RS latch
Other	-	Reserved

This translates to the following code:

```
CCL.SEQCTRL0 = CCL_SEQSEL0_RS_gc;
```

To complete the setup and enable the LUT0 output on the LUT0OUT pin (PA3), the used LUTs and CCL need to be enabled.

```
CCL.LUT1CTRLA |= CCL_ENABLE_bm;
CCL.LUT0CTRLA |= CCL_ENABLE_bm | CCL_OUTEN_bm;
CCL.CTRLA = CCL_ENABLE_bm;
```

Tip: The full code example is also available in the Appendix section.

View the ATmega4809 Code Example on GitHub
Click to browse repository

An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:

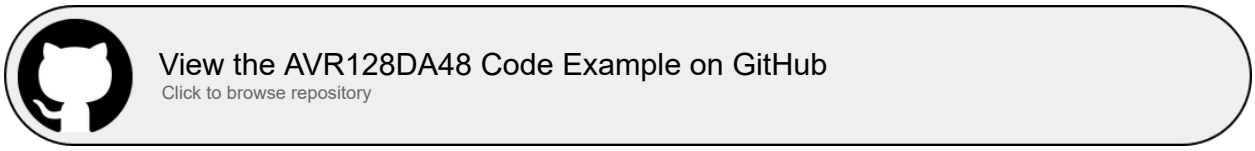
View the AVR128DA48 Code Example on GitHub
Click to browse repository

6. Advanced Examples

This section consists of advanced use cases that use the CCL peripheral combined with other several peripherals. These use cases are generated using MCC and are developed on AVR128DA48.

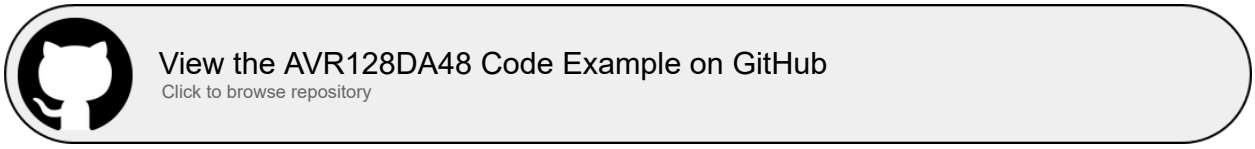
Manchester Encoder

This use case is an implementation of a Manchester encoder using Core Independent Peripherals (CIPs) by following the interaction between CCL, USART and Event System peripherals. The raw data are received via serial communication, encoded using a circuit composed of the CIPs mentioned above, and sent further through a single data wire.



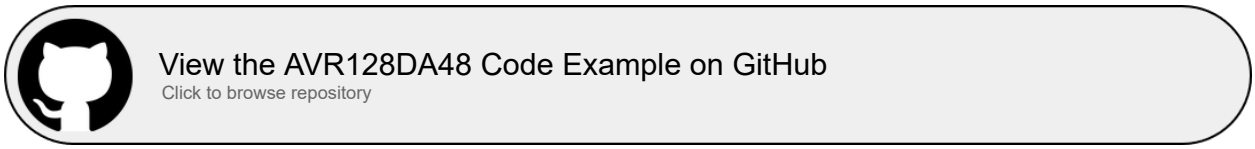
Manchester Decoder

This use case is an implementation of a Manchester decoder using CIPs by following the interaction between CCL, Event System, TCB and SPI peripherals. The encoded data are received through a single data wire. The NRZ (Non-Return-to-Zero) signal and clock signal are recovered using the circuit composed of the CIPs mentioned above. The resulting signals are routed to the SPI peripheral which reads the data. The decoded data are transmitted further via serial communication.



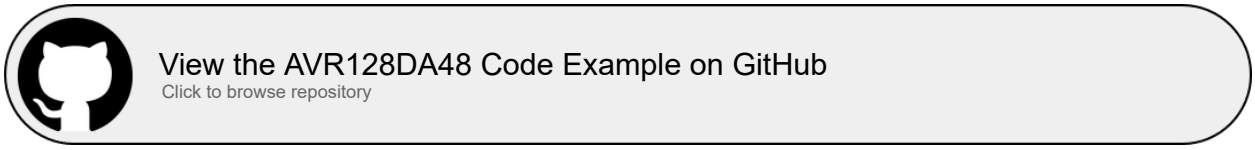
Bi-Phase Encoder

This project is an implementation of a Bi-phase encoder using CIPs by following the interaction between CCL, Event System, SPI and USART peripherals. The raw data are received via serial communication, encoded using the circuit composed of the CIPs mentioned above, and sent further through a single data wire.



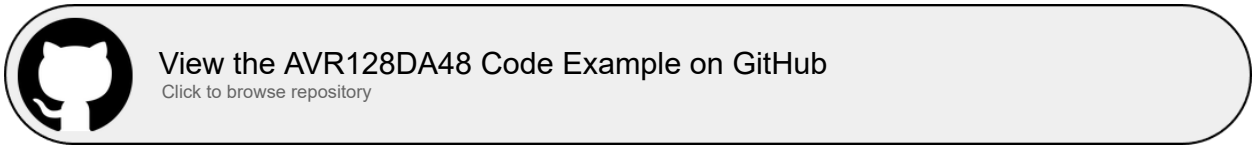
Bi-Phase Decoder

This project is an implementation of a Bi-phase decoder using CIPs by following the interaction between CCL, TCA, TCB, USART, Event System and SPI peripherals. The encoded data are received through a single data wire. The NRZ (Non-return-to-zero) signal and clock signal are recovered using the circuit composed of the CIPs mentioned above. The resulting signals are routed to the SPI peripheral which reads the data. The decoded data is transmitted further via serial communication.



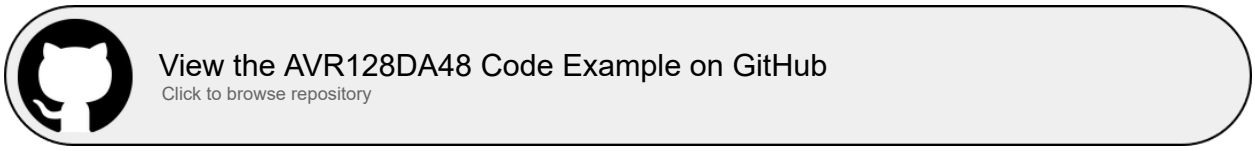
Driving a Metronome

The use case consists of a circuit composed of CIPs, which is capable of creating the signals that drive a Switec stepper motor as a metronome. It also adjusts the number of beats per minute of the metronome by reading an input value provided by the user.



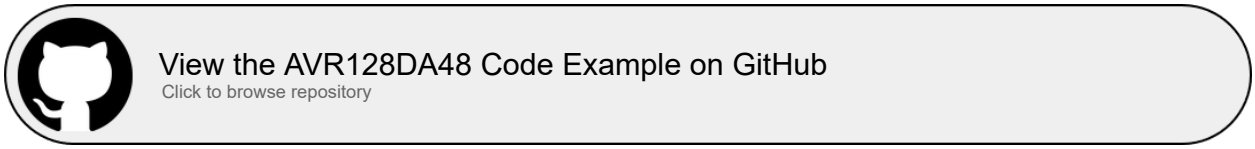
SOS Sequence Generator

This use case consists of a circuit composed of CIPs, which operates with the involvement of the core only in the initialization part and generates an SOS sequence signal.



RGB Lighting with WS2812

This use case consists of a circuit composed of CIPs for interfacing the CCL and SPI peripherals with the WS2812 LED.



7. References

1. ATmega4809 product page: www.microchip.com/wwwproducts/en/ATMEGA4809
2. megaAVR® 0-series Family Data Sheet
3. ATmega809/1609/3209/4809 – 48-Pin Data Sheet megaAVR® 0-series
4. ATmega4809 Xplained Pro product page: https://www.microchip.com/developmenttools/ProductDetails/atmega4809-xpro
5. AVR128DA48 product page: www.microchip.com/wwwproducts/en/AVR128DA48
6. AVR128DA48 Curiosity Nano Evaluation Kit product page: https://www.microchip.com/Developmenttools/ProductDetails/DM164151
7. AVR128DA28/32/48/64 Data Sheet
8. Getting Started with the AVR® DA Family

8. Revision History

Document Revision	Date	Comments
C	06/2022	Added AVR® DB and tinyAVR® 2 to relevant devices.
B	03/2021	Updated the GitHub repository links, the *References* section, the use cases sections, and the use cases figures. Added the *AVR® DA Family Overview*, *Advanced Examples* and *Revision History* sections. Added MCC versions for each use case, running on AVR128DA48. Other minor corrections.
A	05/2019	Initial document release.

9. Appendix

Example 9-1. Logic AND Gate Code Example

```
#include <avr/io.h>

void PORT_init (void);
void CCL_init(void);

/**
 * \brief Initialize ports
 */
void PORT_init (void)
{
    PORTC.DIR &= ~PIN0_bm;          //PC0 - LUT1 IN[0]
    PORTC.DIR &= ~PIN1_bm;          //PC1 - LUT1 IN[1]
    PORTC.DIR &= ~PIN2_bm;          //PC2 - LUT1 IN[2]

    PORTC.DIR |= PIN3_bm;            //PC3 - LUT1 output
}

/**
 * \brief Initialize CCL peripheral
 */
void CCL_init(void)
{

//configure inputs for used LUTs
    CCL.LUT1CTRLB = CCL_INSEL0_IO_gc    /* IO pin LUTn-IN0 input source */
                  | CCL_INSEL1_IO_gc; /* IO pin LUTn-IN1 input source */
    CCL.LUT1CTRLC = CCL_INSEL2_IO_gc;   /* IO pin LUTn-IN2 input source */

//Configure Truth Table
    CCL.TRUTH1 = 0x80; /* Truth 1: 128 */

//Enable LUT0 output on IO pin
    CCL.LUT1CTRLA = CCL_OUTEN_bm;     /* Output Enable: enabled */

//Enable LUTs
    CCL.LUT1CTRLA |= CCL_ENABLE_bm;    /* LUT Enable: enabled */

//Enable CCL module
    CCL.CTRLA = CCL_ENABLE_bm;          /* Enable: enabled */
}


int main(void)
{
    PORT_init();
    CCL_init();
    while (1)
    {
        ;
    }
}
```

Example 9-2. State Decoder Code Example

```
#include <avr/io.h>

void PORT_init (void);
void CCL_init(void);

/**
 * \brief Initialize ports
 */
void PORT_init (void)
{

    PORTA.DIR &= ~PIN0_bm;          //PA0 - LUT0 IN[0]
    PORTA.DIR &= ~PIN1_bm;          //PA1 - LUT0 IN[1]
    PORTC.DIR &= ~PIN0_bm;          //PC0 - LUT1 IN[0]
    PORTC.DIR &= ~PIN1_bm;          //PC0 - LUT1 IN[1]
    PORTC.DIR &= ~PIN2_bm;          //PC0 - LUT1 IN[2]

    PORTA.DIR |= PIN3_bm;             //PA3 - LUT0 output

}

/**
 * \brief Initialize CCL peripheral
 */
void CCL_init(void)
{

//configure inputs for used LUTs
    CCL.LUT0CTRLB = CCL_INSEL0_IO_gc    /* IO pin LUTn-IN0 input source */
                  | CCL_INSEL1_IO_gc; /* IO pin LUTn-IN1 input source */
    CCL.LUT0CTRLC = CCL_INSEL2_LINK_gc; /* Linked LUT input source */

    CCL.LUT1CTRLB = CCL_INSEL0_IO_gc    /* IO pin LUTn-IN0 input source */
                  | CCL_INSEL1_IO_gc; /* IO pin LUTn-IN1 input source */
    CCL.LUT1CTRLC = CCL_INSEL2_IO_gc;   /* IO pin LUTn-IN2 input source */

//Configure Truth Tables
    CCL.TRUTH0 = 0x40; /* Truth 0: 64 */
    CCL.TRUTH1 = 0x20; /* Truth 1: 32 */

//Enable LUT0 output on IO pin
    CCL.LUT0CTRLA = CCL_OUTEN_bm;     /* Output Enable: enabled */

//Enable LUTs
    CCL.LUT0CTRLA |= CCL_ENABLE_bm;    /* LUT Enable: enabled */
    CCL.LUT1CTRLA = CCL_ENABLE_bm;    /* LUT Enable: enabled */

//Enable CCL module
    CCL.CTRLA = CCL_ENABLE_bm;         /* Enable: enabled */
}

int main(void)
{
    PORT_init();
    CCL_init();
    while (1)
    {
        ;
    }
}
```

Example 9-3. SR Latch Code Example

```
#include <avr/io.h>

void PORT_init (void);
void CCL_init(void);

/**
 * \brief Initialize ports
 */
void PORT_init (void)
{

    PORTA.DIR &= ~PIN1_bm;          //PA1 - LUT0 IN[1]
    PORTC.DIR &= ~PIN1_bm;          //PC1 - LUT1 IN[1]

    PORTA.DIR |= PIN3_bm;            //PA3 - LUT0 output

}

/**
 * \brief Initialize CCL peripheral
 */
void CCL_init(void)
{

//configure inputs for used LUTs
    CCL.LUT0CTRLB = CCL_INSEL0_MASK_gc    /* LUTn-IN0 input masked */
                  | CCL_INSEL1_IO_gc; /* IO pin LUTn-IN1 input source */
    CCL.LUT0CTRLC = CCL_INSEL2_MASK_gc; /* LUTn-IN2 input masked */

    CCL.LUT1CTRLB = CCL_INSEL0_MASK_gc    /* LUTn-IN0 input masked */
                  | CCL_INSEL1_IO_gc; /* IO pin LUTn-IN1 input source */
    CCL.LUT1CTRLC = CCL_INSEL2_MASK_gc; /* LUTn-IN2 input masked */

//Configure Truth Tables
    CCL.TRUTH0 = 0x01; /* Truth 0: 1 */
    CCL.TRUTH1 = 0x01; /* Truth 1: 1 */

// Configure filter
    CCL.LUT0CTRLA = CCL_FILTSEL_FILTER_gc;      /* Enable filter*/
    CCL.LUT1CTRLA = CCL_FILTSEL_FILTER_gc;      /* Enable filter*/

//Enable sequential logic for LUT0 and LUT1
    CCL.SEQCTRL0 = CCL_SEQSEL0_RS_gc;

//Enable LUT0 output on IO pin
    CCL.LUT0CTRLA |= CCL_OUTEN_bm;     /* Output Enable: enabled */

//Enable LUTs
    CCL.LUT0CTRLA |= CCL_ENABLE_bm;    /* LUT Enable: enabled */
    CCL.LUT1CTRLA |= CCL_ENABLE_bm;    /* LUT Enable: enabled */

//Enable CCL module
    CCL.CTRLA = CCL_ENABLE_bm;         /* Enable: enabled */
}


int main(void)
{
    PORT_init();
    CCL_init();
    /* Replace with your application code */
    while (1)
    {
        ;
    }
}
```

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded

by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet- Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, NVM Express, NVMe, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-6683-0539-3

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820