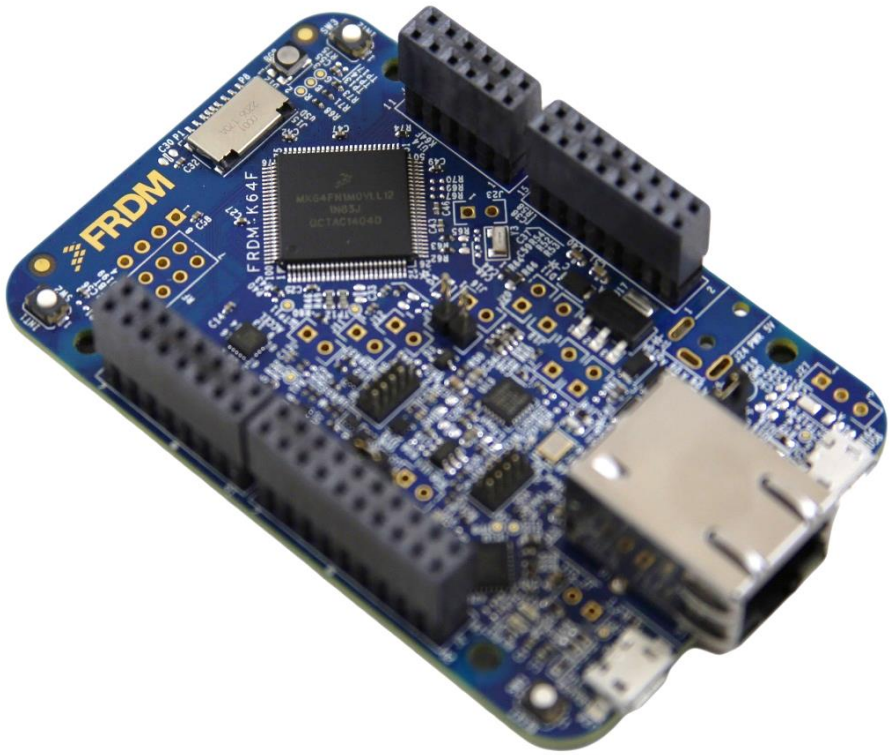
# What is and how to configure the eDMA channel linking feature

By: Technical Information Center

## Introduction

The Enhanced Direct Memory Access (eDMA) is a module capable of performing complex data transfers with minimal intervention from a host processor. This peripheral has a lot of possible configurations and modes of operation that increment the possibilities to automatize the data transfers. This module has a feature called channel linking which is explained in this document.




freescale semiconductor

# Contents

## 1. About this document

This document explains the channel linking feature which is present in the Enhanced Direct Memory Access (eDMA) peripheral. If you need a closer view to this peripheral I widely recommend you to watch the eDMA training video or read the documents "Using the DMA module in Kinetis Devices" and "What is and how to configure the eDMA scatter/gather feature".

## 2. eDMA channels

Each eDMA module contains 16 channels which are used to perform data transactions without the CPU intervention. Each channel contains its own Transfer Control Descriptor (TCD) which means that each channel has its own configuration.

A common use case is to configure one channel to perform one simple activity in an entire application which means that a channel is usually configured to have the same source and the same destination in the whole execution (data from one peripheral to a buffer or vice versa). Also, it is usually necessary to perform two or more concurrent transactions with different channels, however, the eDMA has one big limitation, although there are a lot of channels only one transaction can be done at the same time. So, what happen if my application needs to perform two or more transactions at the same time? The answer to this problem could be the eDMA channel linking.

## 3. Channel linking feature

The channel linking or channel chaining is a mechanism where one channel initiates a service request to another channel (i.e. the bit TCDn_CSR[START] is set) when the first channel has terminated its minor or its major loop, depending on the configuration.

This doesn't allow an application to perform two or more concurrent transactions, but this does allow a channel to start one transaction when another one finishes its execution which would solve the concurrency problem. The best of this feature is that there is not CPU intervention!

## 4. Channel linking on minor loop complete

As the channel completes the minor loop, the flag DMA_TCDn_CITER[ELINK] enables the linking to another channel which is defined by the DMA_TCDn_CITER[LINKCH] field. The link target channel initiates a channel service request via an internal mechanism that sets the DMA_TCDn_CSR[START] bit of the specified channel.

When enabled, the channel link is made after each iteration of the minor loop except for the last (the last should be configured as channel linking on major loop complete).

It is important to know that when the software loads the Transfer Control Descriptor (TCD), the register DMA_TCDn_CITER must be set equal to the register DMA_TCDn_BITER; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of the BITER register are reloaded into the CITER one.

When this feature is disabled, the DMA_TCDn_CITER[CITER] field uses a 15-bit vector instead of a nine one to form the current iteration count. The bits associated with the DMA_TCDn_CITER[LINKCH] are concatenated onto the CITER value to increase the range of the CITER.

## 5. Channel linking on major loop complete

As the channel completes the major loop, the flag DMA_TCDn_CSR[MAJORELINK] enables the linking to another channel which is defined by the DMA_TCDn_CSR[MAJORLINKCH] field. The link target channel initiates a channel service request via an internal mechanism that sets the DMA_TCDn_CSR[START] bit of the specified channel.

Both the channel linking on the minor and the major loop can be enabled for the same channel.

## 6. Use case

Let's imagine an audio application like an equalizer. It makes sense to configure one channel to move data from an Analog-to-Digital Converter (ADC) to a ping pong buffer and another to move data from the buffer to a Digital-to-Analog Converter (DAC). Also, there must be a base time which can be generated with a timer capable to start the ADC conversions, let's say every 50 us (20 KHz), like the Periodic Interrupt Timer (PIT), and configure the ADC to generate eDMA trigger events when a conversion is finished. This would result in an application where the eDMA makes all the transactions to keep the CPU free to make all the audio data processing, in this case, to filter the signal.

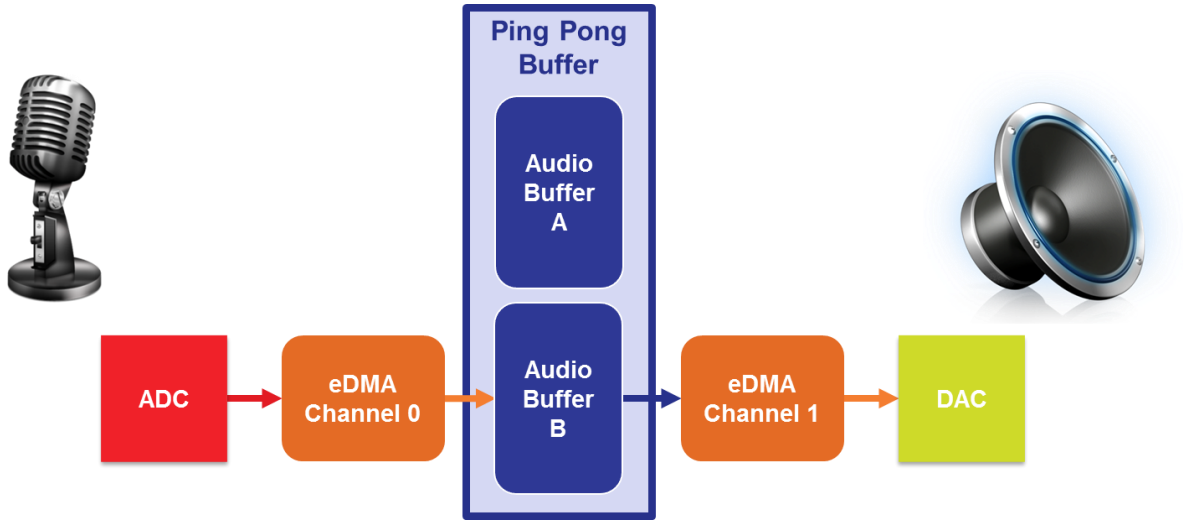



*Figure 1. Audio equalizer block diagram.*

The first idea could be to trigger both transactions at the same time but as we previously said, this is not possible. Since the sampling frequency should be the same in the input and in the output, a channel linking scheme is an alternative to perform these tasks.

The proposed solution is to create a base time with the PIT every 50 us and configure the ADC to start converting with this hardware trigger. Once the conversion is finished, it could be configured as the eDMA trigger source for the channel 1 which will move a processed sample from the buffer to the DAC. Through the channel linking feature, when the processed sample, which is still stored in the buffer, has been converted by the DAC, the channel 0 is triggered to store the result of the conversion in the buffer.

## 7. Conclusion

This document has demonstrated the utility of the channel linking feature which is present in the eDMA. This is a very powerful feature because it solves the main eDMA's constrain that is to use only one channel at the same time through a simple logic between linked channels without CPU processing.