

EFM[®]32

... the world's most energy friendly microcontrollers

Direct Memory Access

AN0013 - Application Note

A decorative graphic on the right side of the page consists of several concentric circles. The outermost circle is light green and contains the number '0'. The next circle inward is a darker green and contains the number '1'. The third circle is blue and contains the number '2'. The fourth circle is a darker blue and contains the number '3'. The innermost circle is black and contains the number '4'. The circles are arranged in a way that they appear to be part of a larger, stylized '0' or '4' shape.

Introduction

This application note demonstrates how to use the Direct Memory Access (DMA) module of the EFM32.

Memory to memory, peripheral to memory, ping-pong and scatter-gather modes are used in the attached software example.

This application note includes:

- This PDF document
- Source files (zip)
 - Example C-code
 - Multiple IDE projects

1 DMA Theory

1.1 General

DMA is used for data transfer without CPU intervention. Data can be transferred between any locations within the EFM32 address space and additionally several peripherals have DMA support. This means that each time new data is to be read or written, the DMA can be triggered to perform a new transfer. The DMA makes the CPU available for other tasks rather than data transfer, or the CPU can enter energy mode 1 (EM1) in order to save energy while data is transferred. Upon completion the DMA can wake up the CPU using interrupt.

1.2 DMA Channels

The EFM32 features 8 DMA channels which may be individually configured. For example, channel 0 may be used to move data received by the I2C to ram, channel 2 may move conversion data from the ADC to ram, while channel 3 transfers data from flash to the EBI etc...

For a complete list of peripherals capable of generating DMA triggers please refer to the DMA section in the reference manual.

1.3 Setup

For each DMA channel the source/destination address, transfer length and other parameters must be specified. The details are described in the subsequent chapter.

2 DMA Channel Configuration

2.1 Transfer Setup

In order for a DMA transaction to take place, several parameters such as source/destination address and transfer length must be specified. The configuration is partly stored in a descriptor stored in RAM. The location of this descriptor must be aligned to a 256 byte-location, i.e. the lower 8 bits of its address must be 0. Please refer to the attached software example for details on how to align the structure.

2.1.1 Source/Destination Address, Size, Length and Increment

The core parameters of the DMA transfer are the source and destination addresses which specifies from which location the data is to be fetched and to where it is to be transferred. The size parameter specifies the size of the transfer unit, i.e 1, 2 or 4 bytes. Additionally, the number of such unit transfers the DMA is to perform (length) must be set. The increment is set individually for both the source and destination address, and indicates whether the address is to be incremented after transferring each unit, and if so, by how much. E.g. when transferring RX data from the USART, the source address is not to be incremented, as the data is read from the same location each time. The destination address, however, must be incremented (e.g. by 1 byte). The transfer size would in this case also be 1 byte.

2.1.2 Transfer Mode

The DMA can operate in several different modes as specified in Table 2.1 (p. 3) .

Table 2.1. DMA Transfer Modes

Mode	Description
Basic	This mode is generally used when transferring data to or from a peripheral. The peripheral asserts a DMA trigger each time new data is available or needed, and in response the DMA performs one unit transfer (e.g. transfers one byte to the USART_TXDATA register). Such unit transfers will be repeated until the number of transfers equals the specified transfer length.
Auto	This mode is used when both the source and destination location is ready to transfer all the data, e.g. when transferring data from flash to ram. Once the transfer is started, the number of transfers specified by the transfer length is performed without waiting for any further triggers.
Ping-Pong	Each Channel has one primary and one alternate structure. In pong-pong mode, the DMA first uses the primary descriptor and then switches to the alternate structure. This mode is useful when it is required that there is an active DMA at all times, e.g. when fetching data from the ADC running at 1Msampe/s.
Scatter-Gather	In this mode the primary DMA descriptor is used to load the alternate DMA descriptor. The DMA will first load the alternate structure and then perform the transfer specified by the alternate structure before loading a new alternate structure which is subsequently executed. In this manner a sequence of transactions can be specified and performed without any run-time intervention by the CPU.

For further details on the modes please refer to the DMA section in the reference manual.

2.1.3 DMA Trigger

When transferring data to or from a peripheral unit, the DMA trigger signal from the peripheral must be specified for the DMA channel to be used. For example, the following code configures DMA channel 0 to use USART0 TX as trigger:

```
DMA->CH[0].CTRL = DMAREQ_USART0_TXBL
```

2.1.4 DMA Interrupt

After a DMA channel has performed the "length" transfers, the DMA channel is done. Upon completion, the DMA channel may generate an interrupt request. If desired the DMA channel can be re-enabled within the DMA interrupt routine.

Note

The DMA controller maintains some of its variables such as transfer length in the DMA descriptor structure in RAM. When re-enabling the DMA, the transfer-length must be set to its correct value.

2.1.5 DMA Priority

Since the CPU and the different DMA channels operate on the same bus structure, contention is likely to occur. There are two ways to prioritize a DMA channel. Firstly, the DMA channel number sets a priority in that channel 0 has the highest priority and channel 7 has the lowest. Additionally, it is possible to set the channel priority to high. A channel with high priority is prioritized over any channels with normal priority. The channel 0 will get the highest possible priority if its priority is set to high.

For further details on the DMA controller and different settings please refer to the included software and the reference manual.

2.1.6 Further Info

For further details on the DMA controller and different settings please refer to the included software and the reference manual.

3 Software Example

3.1 General

The attached software example demonstrates how to use the DMA in different modes. In the sections below more details about each example-stage are given.

While waiting for a DMA channel to complete, Sleep Mode (EM1) is entered in order to save energy.

3.1.1 Flash Transfer

The first example illustrates how to use the DMA to transfer data between to memory locations. A set of data located in flash is transferred to RAM using the DMA. In this transfer, the Auto setting is used, which makes the DMA transfer the whole buffer once it has started. Upon completion the call-back function is called.

3.1.2 ADC Transfer

This example demonstrates how to use the DMA to transfer data from a peripheral to RAM. The DMA channel is set up to use the Basic mode, which makes the DMA transfer one unit each time the peripheral (i.e. the ADC in this example) sets its DMA trigger. When the specified number of units have been transferred, the corresponding call-back function is called.

3.1.3 ADC Transfer (Ping-Pong)

This example differs from the previous one in that the DMA uses Ping-Pong mode. The transfer is re-enabled a number of times in order to demonstrate how to refresh the DMA. After both the primary and the alternate DMA structure has been utilized a defined number of times, the example stage is terminated by not updating the DMA structures.

3.1.4 Scatter-Gather Transfer

In this example a sequence of 3 transfers is executed using the Scatter-Gather mode. The primary structure is used to sequentially load the alternate structure with the 3 core transfers to be performed. When the third transfer is completed, the call-back function is called and the example is terminated by entering an eternal while-loop.

4 Revision History

4.1 Revision 1.04

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

4.2 Revision 1.03

2012-03-14

Fixed makefile-error for CodeSourcery projects.

4.3 Revision 1.02

2011-03-24

Changed some incorrect function descriptions in dmaexample.c .

4.4 Revision 1.01

November 16th, 2010.

Changed example folder structure, removed build and src folders.

Added chip-init function.

4.5 Revision 1.00

September 20th, 2010.

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Energy Micro AS intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Energy Micro products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Energy Micro reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Energy Micro shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Energy Micro. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Energy Micro products are generally not intended for military applications. Energy Micro products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

A.2 Trademark Information

Energy Micro, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Energy Micro AS. ARM, CORTEX, THUMB are the registered trademarks of ARM Limited. Other terms and product names may be trademarks of others.

B Contact Information

B.1 Energy Micro Corporate Headquarters

Postal Address	Visitor Address	Technical Support
Energy Micro AS P.O. Box 4633 Nydalen N-0405 Oslo NORWAY	Energy Micro AS Sandakerveien 118 N-0484 Oslo NORWAY	support.energymicro.com Phone: +47 40 10 03 01

www.energymicro.com

Phone: +47 23 00 98 00

Fax: + 47 23 00 98 01

B.2 Global Contacts

Visit **www.energymicro.com** for information on global distributors and representatives or contact **sales@energymicro.com** for additional information.

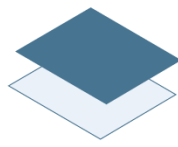
Americas	Europe, Middle East and Africa	Asia and Pacific
www.energymicro.com/americas	www.energymicro.com/emea	www.energymicro.com/asia

Table of Contents

- 1. DMA Theory 2
 - 1.1. General 2
 - 1.2. DMA Channels 2
 - 1.3. Setup 2
- 2. DMA Channel Configuration 3
 - 2.1. Transfer Setup 3
- 3. Software Example 5
 - 3.1. General 5
- 4. Revision History 6
 - 4.1. Revision 1.04 6
 - 4.2. Revision 1.03 6
 - 4.3. Revision 1.02 6
 - 4.4. Revision 1.01 6
 - 4.5. Revision 1.00 6
- A. Disclaimer and Trademarks 7
 - A.1. Disclaimer 7
 - A.2. Trademark Information 7
- B. Contact Information 8
 - B.1. Energy Micro Corporate Headquarters 8
 - B.2. Global Contacts 8

List of Tables

2.1. DMA Transfer Modes 3



ENERGY[®]
micro

*Energy Micro AS
Sandakerveien 118
P.O. Box 4633 Nydalen
N-0405 Oslo
Norway*

www.energymicro.com