EXAMPLE NAME: XMC4500_Relax_Kit_DMA_FLASH_SRAM

OVERVIEW:
This example shows the usage of the DMA_CH APP.The DMA_CH APP it is used to transfer data between memory to memory, memory to peripheral, peripheral to memory or peripheral to peripheral without the intervention of the CPU.

DESCRIPTION:
In this example, the DMA_CH APP is used to transfer data from flash to SRAM. The example shows the advantage of using the DMA for such a task instead of copying using the CPU.
By defining MEMCPY macro different than zero, the CPU will be used to copy the data suing the standard
library function memcpy. Otherwise the copying of data is done using the DMA.
Either method is timed suing the SysTick:
a) CPU method

28963 CPU cycles at 120MHz, -241us

b) DMA method

9785 CPU cycles approximately at 120MHz, i.e. -81us with Optimization O1

In addition the CPU is free during copying time to do other activities, and will get notified if desired when the copy is done.

REQUIRMENTS:
Boards Required: XMC4500 Relex Kit Board (Order Nr: KIT_XMC45_RELAX_V1)

HOW TO CREATE THE PROJECT:

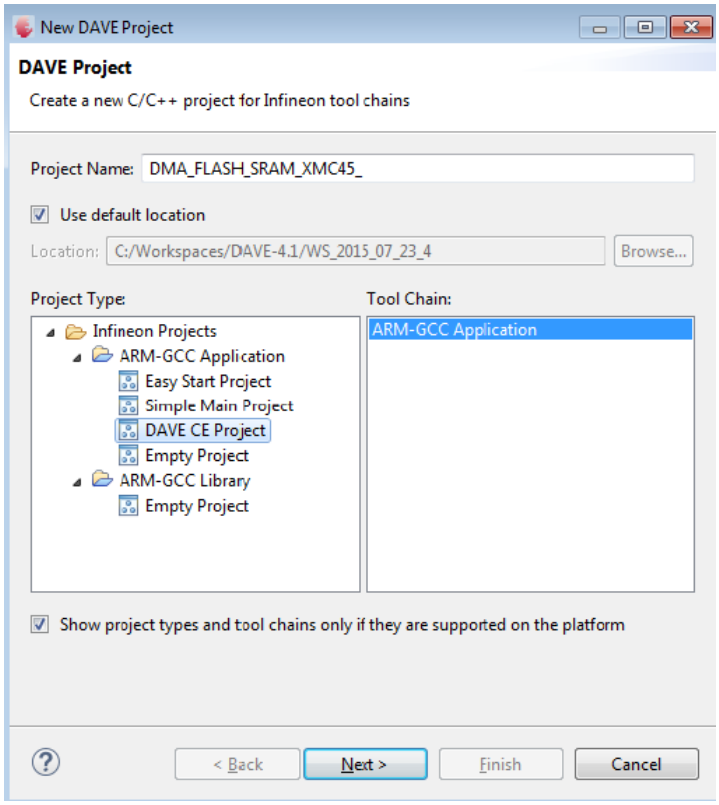
1. Create a new DAVE CE Project
To create new projects go to **File** menu, select **New** and choose **DAVE CE Project**.

In the **New DAVE Project** window:

- in the Project name: field enter the name of the new project, i.e. XMC4500_Relax_Kit_External_Interrupt.
- in the Project type: section select **DAVE CE Project**.
- in the Toolchains: section select **ARM-GCC**

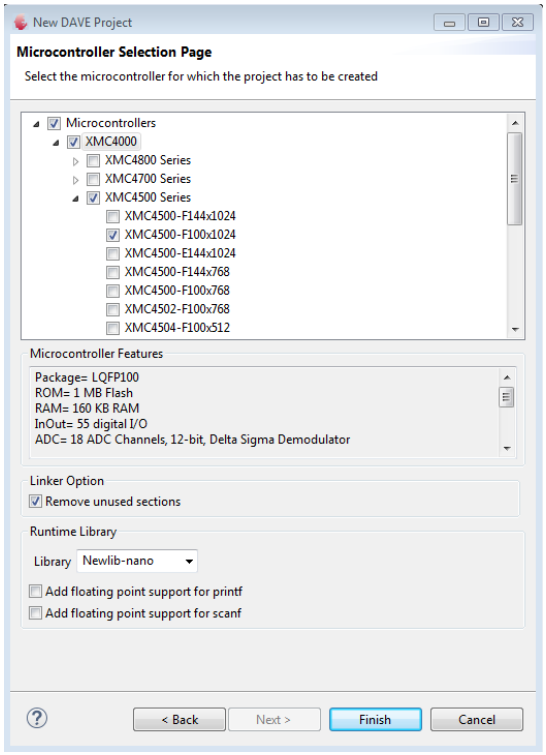
Application

- click the Next > button



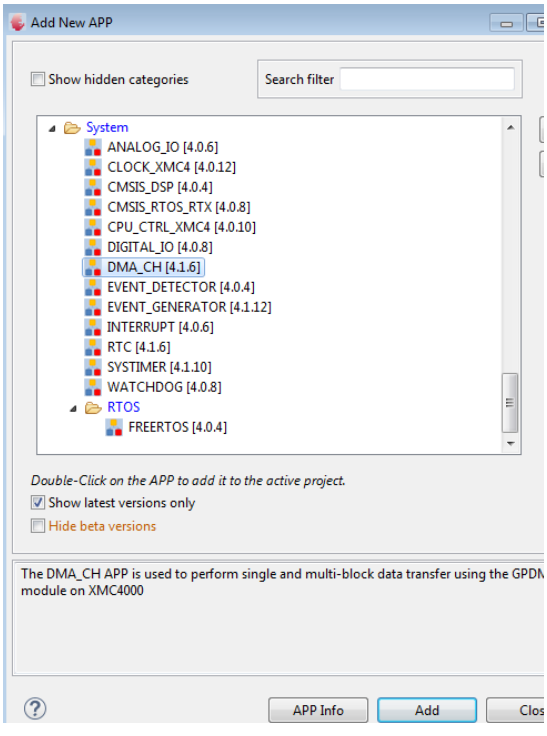
In the **Target Selection Page**:

- select the target device, i.e. **XMC4500-F100x1024**
- click the **Finish** button


New DAVE Project
Microcontroller Selection Page
Select the microcontroller for which the project has to be created
Microcontrollers
XMC4000
XMC4800 Series
XMC4700 Series
XMC4500 Series
XMC4500-F144x1024
XMC4500-F100x1024
XMC4500-E144x1024
XMC4500-F144x768
XMC4500-F100x768
XMC4502-F100x768
XMC4504-F100x512
Microcontroller Features
Package= LQFP100
ROM= 1 MB Flash
RAM= 160 KB RAM
InOut= 55 digital I/O
ADC= 18 ADC Channels, 12-bit, Delta Sigma Demodulator
Linker Option
Remove unused sections
Runtime Library
Library
Newlib-nano
Add floating point support for printf
Add floating point support for scanf
< Back
Next >
Finish
Cancel

2. Add APPs to your project

Use the *Add New APP* button in toolbar or go to **DAVE** menu and select *Add New APP*. In the **Add New APP** dialog, add the following APPs by double clicking:



- add a DMA_CH APP
- click the **Close** button

3. Configure APPs and signals

. Double click on DMA_CH_0 APP instance to configure the APP:

In the Channel Settings tab:

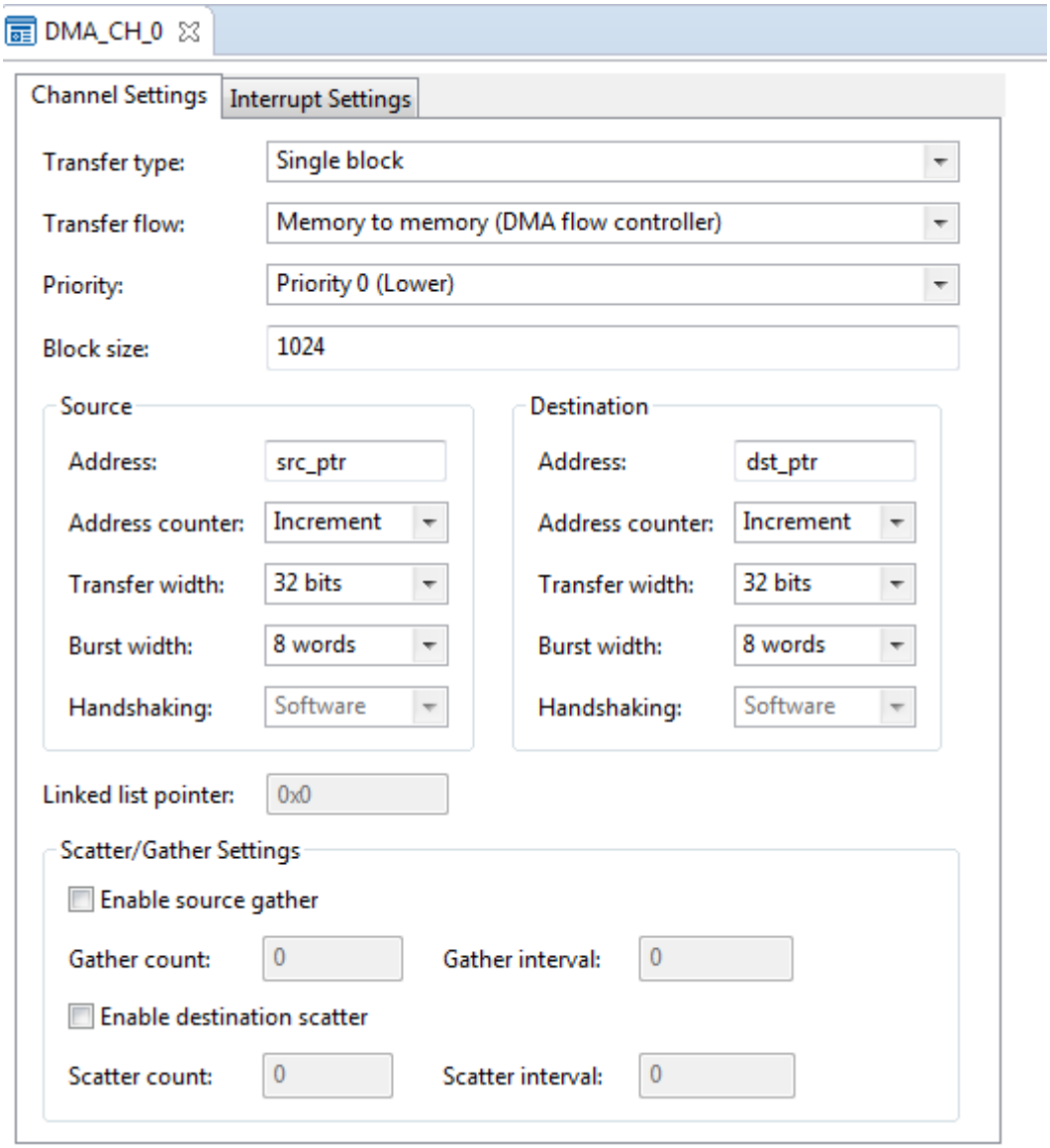
- Select Single block in Transfer type: selection
- In this example a single block of data is transferred. After the transfer is complete the DMA channel will be disabled
- Select Memory to memory (DMA flow controller) in Transfer flow: selection
- Memory to memory is chosen as the transfer occurs between flash and SRAM. In general by memory is meant any peripheral that does not require handshaking and therefore it is always IO ready.
- Select Priority 0 (Lower) in Prioriry: selection
- No other DMA channel is configured in this example, therefore is the priority is not relevant.

- enter **1024** in Block Size:
- This is the size of the data transfer.
- enter **src_ptr** in Source Address:
  This is the Start address in FLASH of data to be transferred. The content of the pointer needs

  to be initialized.
- select **Increment** in Source Address counter: selection
- The internal DMA source address pointer is to be incremented after every single word transfer
- select **32 bits** in Source Transfer width: selection
- This yields the best performance when transferring data from memory.
- select **8 words** in Source Burst width: selection
- DMA0 CH0 and CH1 have a FIFO of 8 words. By this setting we can store a complete burst in the DMA and the bus utilization is minimized. The DAVE solver will use this information to assign the correct hardware resource to the DMA_CH APP, i.e. DMA0 CH0 or CH1.
- enter **dst_ptr** in Destination Address:
  This is the start address in SRAM where the data is to be copied. The content of the pointer

  need to be initialized by the user in the code
- select **Increment** in Destination Address counter: selection
- The internal DMA destination address pointer is to be incremented after every single word transfer
- select **32 bits** in Destination Transfer width: selection
- This yields the best performance when transferring data from memory.
- select **8 words** in Source Burst width: selection

  DMA0 CH0 and CH1 have a FIFO of 8 words. By this setting we can store a complete burst in the DMA and the bus utilization is minimized. The DAVE solver will use this information to assign the correct hardware resource to the DMA_CH APP, i.e. DMA0 CH0 or CH1.


DMA_CH_0
Channel Settings
Interrupt Settings
Transfer type:
Single block
Transfer flow:
Memory to memory (DMA flow controller)
Priority:
Priority 0 (Lower)
Block size:
1024
Source
Address:
src_ptr
Address counter:
Increment
Transfer width:
32 bits
Burst width:
8 words
Handshaking:
Software
Destination
Address:
dst_ptr
Address counter:
Increment
Transfer width:
32 bits
Burst width:
8 words
Handshaking:
Software
Linked list pointer:
0x0
Scatter/Gather Settings
Enable source gather
Gather count:
0
Gather interval:
0
Enable destination scatter
Scatter count:
0
Scatter interval:
0

4. Generate the code

Use the button  in toolbar or go to **DAVE** menu and select Generate Code.

In the project a new folder Generated is created under DAVE folder. This folder contains a subfolder for every APP type used in the project. Each APP subfolder contains the generated APP configuration in the form of configuration structures, i.e. DMA_CH_conf.c. In addition, the APP API library files are copied.

5. Add user code to main.c (in parenthesis is the line number)

(2) Add string.h to use memcpy
(7-137) Add the data to be copied. Notice the const modifier to locate the data in FLASH
(140) Define the destination array
(143-146) The pointers referenced in the UI need to be initialized.
(148-146) These are the time variables and routines to do the time measurement
(191-193) A calibration of the time measurement routine is done
(195-208) The timer is started, and the copy is done using one of the methods.
6. Build and download to the microcontroller.